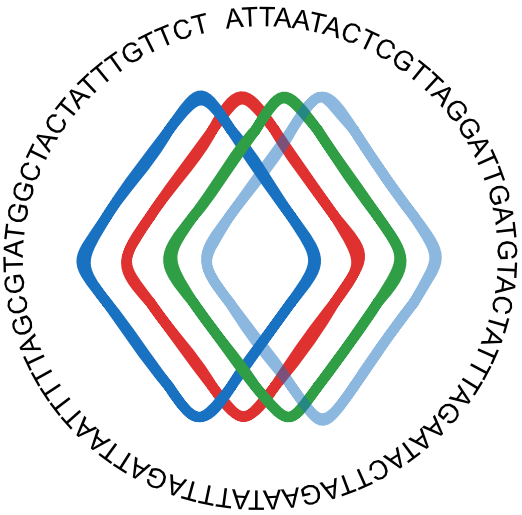
Introduction to Bioinformatics



Ebrahim Afyounian

2023

Last updated: 22.11.2023

⁂

Table of Contents

Chapter 1. Computers and Unix-line operating systems

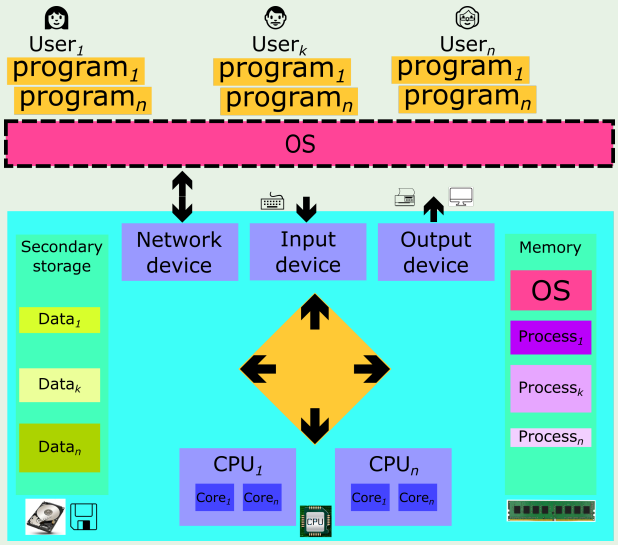
In this chapter, we briefly introduce some concepts related to computers. We provide links to different resources such as Wikipedia entries for those who are interested to learn more.

The components of the current computers

Modern computers are composed of **several key components**, such as the **central processing unit (CPU)** and **random access memory (RAM)**, which we will review here. **Figure 1** provides a visual depiction of these primary elements.

Figure

Figure 1. An abstract view of the current computers



Central processing unit (CPUs)

- The CPU is the brain of the computer and its job is to **execute computer programs**.
  - Programs are made up of a series of **operations** (aka. **instructions**), encompassing arithmetic, logic, controlling, and input/output (I/O) operations.
- A typical CPU has the following **three parts**:
  - A control unit that decodes and interprets instructions, manages the other parts of the CPU. Additionally, it controls the flow of data between the CPU

    and RAM.
    - An arithmetic logic unit (ALU) that performs arithmetic (e.g., `1+1`) and logical operations (e.g., `a < 0`) on data.
    - A fast data storage called cache which is faster than normal memory but it is more expensive to design. The cache is faster because sits closer to the CPU than the RAM so it takes less time from the data to reach the CPU. If there is a piece of data is used often or a result of an earlier computation that may be needed later, this can be kept in cache to increase the speed.
- Modern computers can have one or more CPUs.
- Each CPU may contain one or more cores. Each core has the ability to independently execute instructions allowing various processes or tasks to run concurrently on multiple cores, enabling simultaneous execution.
    - If there are more processes than there are cores (which is often the case), then the operating system (OS) manages the situation by alternating over different processes and allocating the CPU time to each of them (this itself needs CPU time).

Random access memory (RAM)

- The data and the instructions used by the CPU for executing a particular program are loaded from a secondary storage device and are stored in the random-access memory (RAM) prior to execution.
    - For each program there can by one or more processes in the RAM.
- The term **random** in a RAM is because we can access any memory location, at any time, and in a random order.
- The content of the RAM is considered **volatile** as its content is erased when power is disconnected, thus, the need for the secondary storage devices.
- The RAM is faster but more expensive than the **secondary storage equipment**. Thus, RAM's capacity is limited and smaller than the secondary storage devices.
    - In the event that RAM gets full or a program needs more memory resources than is available, the OS use part of a secondary storage device to keep part of the RAM content. These content are swapped back and forth between the secondary storage device and the RAM via a process called memory paging or swapping.

Secondary/auxiliary storage

- Secondary storage devices, like **hard disk drives** or **compact disks**, exhibit a distinctive characteristic compared to RAM – they retain data even in the absence of power. This characteristic classifies them as **persistent storage**.
- They are slower than RAM because random access is not possible.

- Solid-state drives (SSD) does not rely on mechanical components thus can access data in a random fashion. However, they are still slower than the RAM.

Operating system (OS)

- The Operating system (OS) plays a crucial role in **managing a computer's resources**, including the CPU, RAM, input/output (I/O), and network connections. It is responsible for efficiently allocating these resources among various programs and processes.
  - In the figure, this is shown by drawing the OS box in between the programs (or software) started by different users and the underlying hardware (e.g., CPU, RAM, I/O devices).
- The OS is essentially a program, which is why it is loaded in the computer's RAM immediately when the system starts, a process commonly referred to as booting.
  - Because the OS is responsible for supervising all programs and resource allocations, it is endowed with **special privileges**.

Input/output (I/O) devices

- Input/output (I/O) devices provide the capability of communication between the computer and the the external environment e.g., a user.

Supercomputers

As we will see in chapters 2 and 3, the amount of biological data is increasing. To be able to analyze it, we need more and more computing power (i.e., for some purposes the computer power offered by a laptop is not enough). One possibility is to use computer clusters or supercomputers. A *computer cluster* is a single system composed of a many computers (a.k.a. **nodes**) working together seamlessly. Crudely speaking, a supercomputer is a big computer cluster. For example, at Tampere University, researchers can currently use a computer cluster called Narvi. Narvi computer cluster has many nodes of different types e.g., it has `140` *CPU-only nodes* with `3000+` CPU cores. As an example of supercomputer, we can mention LUMI which is located in Kajaani, Finland with a computation power equal to `1.5` **million modern laptops**.

To be able to use these systems, we need to apply for access. Once the access has been granted, we need to use our computers to connect to them remotely e.g., using a secure shell. These systems, by default, use a **Unix-like operating system** such as Linux. In this chapter, we learn a bit more about Unix commands and tools which prepares us for working with these systems in the future.

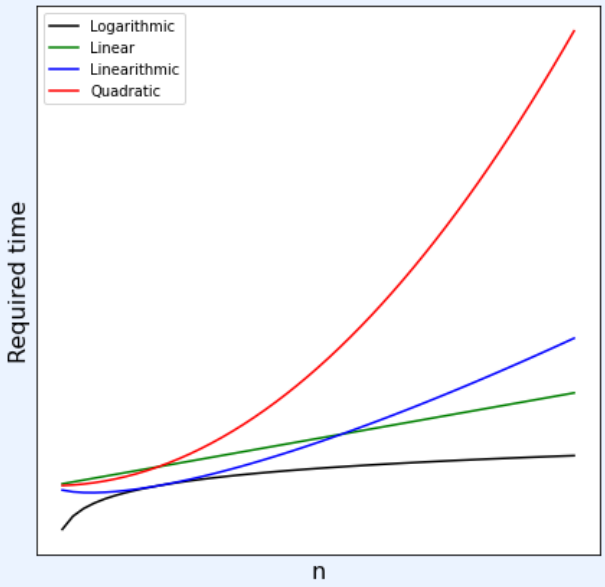
Algorithms and flowcharts

An algorithm is a step by step list of directions used to solve a problem.

Note

When designing algorithms, we are almost always interested to know how much time (or other resources such as memory) it requires to complete. Obviously, we often desire/prefer to create/use a more **efficient** algorithm that solves a problem faster (also using less of other resources). In computer science, the task of calculating this time is known as calculating the *algorithmic complexity* (aka *computational complexity*).

There are different categories of algorithms according to their time requirement. The following figure shows a few different categories. Note how different categories of algorithms require different amount of time as the input size (*n*) to the algorithm increases.



Question

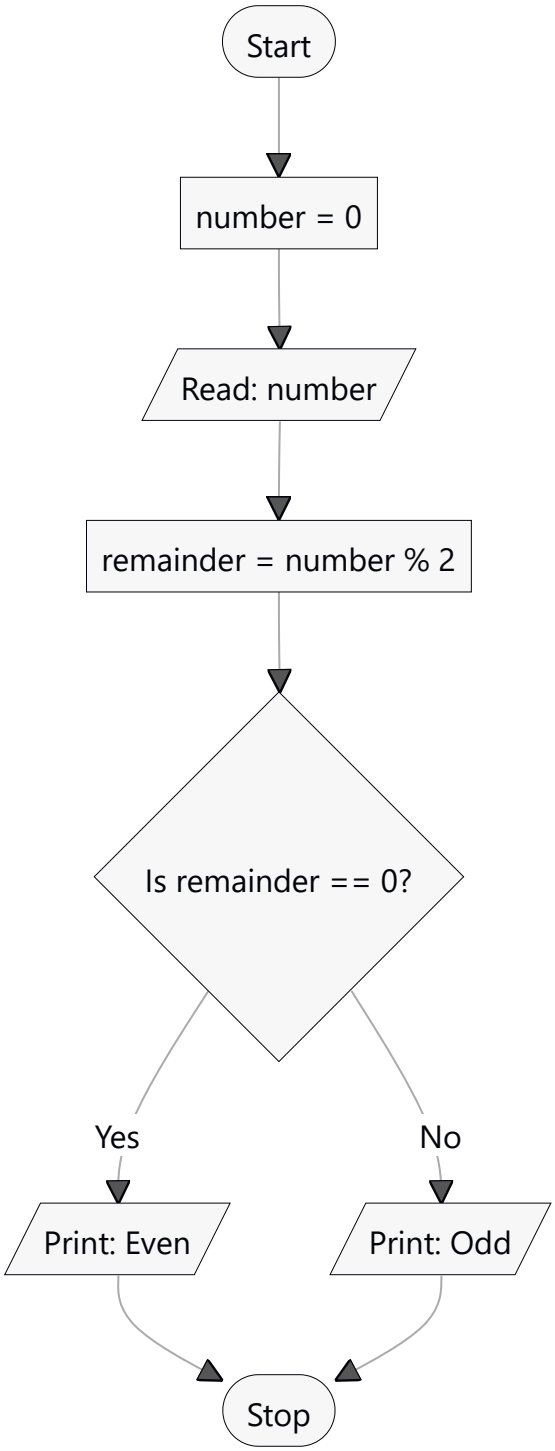
Later, we learn about DNA sequencing. The current high-throughput sequencing machines find the sequence of millions of DNA fragments in parallel and output the sequences. These outputs are known as **sequencing reads** or simply **reads**. Assuming that we have a reference genome available, often the next step, is to find out where in the reference genome these DNA fragments/reads are originating. This task is known as a **sequence alignment** or simply **alignment**. The tool which performs this task is called an **sequence aligner** which uses an algorithm under the hood. Assume that we have 10^9 reads to align. Based on the

figure above which one of these algorithmic categories requires the least amount of time to complete the task of aligning the reads?

Also, think what happens if, thanks to technological development, we get 10^{12} reads from sequencing which is `3` orders of magnitude larger. How does these different categories scale up?

Note. It is possible that for a given task, we may not be able to design an algorithm that belongs to a desired category of algorithms.

A flowchart can be used to visually represent an algorithm using boxes and arrows. For example, let us look at the following flowchart for deciding whether a number is odd or even.


Start
number = 0
Read: number
remainder = number % 2
Is remainder == 0?
Yes
No
Print: Even
Print: Odd
Stop

Programming languages

Once we have crafted an **algorithm** tailored to a specific purpose, the next step is its **implementation** through a **programming language**. Computer processors inherently **understand binary instructions**, expressed as **streams of 0s and 1s**, commonly known as **machine language** or **machine code**. In the early era of computing, individuals had to painstakingly implement their algorithms using machine code, which meant directly working with these binary digits.
In 1950s, more human-readable language such as **assembly** emerged. Rather than writing sequences of 0s and 1s, programmers could use simple names, also referred to as mnemonics, to convey instructions to the CPU. The program written in assembly required an intermediate helper program called an **assembler** to read these text-based instructions and convert it to binary instructions understandable by the CPU.

Despite this advancement, programmers still needed to keep track of many details (e.g., knowing the memory location where a piece of data was residing). Over time, a plethora of more human-readable programming languages emerged alleviating the burden of managing low-level details, allowing programmers to concentrate more on the essence of their programs. Nevertheless, computer programs written in these human-readable languages must ultimately be transformed into machine code, which is done by programs called **compilers**.

> **Info**
>
> Here is a list of human-readable computer languages categorized by the decade they first appeared, although it's worth noting that some are more human-readable than others.
>
> - **1950**. Fortran, COBOL,
> - **1960s**. ALGOL, LISP, BASIC,
> - **1970s**. PASCAL, C, SMALLTALK,
> - **1980s**. C++, Objective-C, PERL,
> - **1990s**. Python, Ruby, Java, R,
> - **After 2000s**. Swift, C#, GO, Julia, Rust.

Programming languages are typically categorized into two main groups: **high-level** and **low-level languages**.

High-level programming language

- High-level programming languages are closer to the natural languages and thus more readable by humans. It is also easier to write code because certain tasks are taken care of by the language itself. Examples are `Python` and `R`.
- The programs written in these languages need to be transformed or converted to instructions that can be understood by a computer's CPU.
- Many of the available bioinformatics tools are written in `Python` or `R`.

Low-level programming languages

- Low-level programming languages are designed to be easy for a computer to execute but more difficult for humans to read.
- Languages such as `C` or `C++` fall in this category but they are still more readable than the following two subcategories. They, however, require that the programmer pay attention to and deal with certain details (e.g., memory management) that are automatically taken care of by high-level languages such as `Python`.
- Except for the **machine code** subcategory, they also need to be converted to instructions that can be understood by a computer's CPU.
- The two less human-readable subcategories are:
  - Assembly language is the second to last in terms of being a low-level language. There are tools called **assemblers** that transform assembly programs to machine code (see below for an example assembly code).
  - Machine code which consists of instructions that directly control a computer's CPU, thus there is not need for conversion.
- Low-level languages have better performance than the high-level languages.

Implementations of the odd/even algorithm

- Let us look at the implementation of the odd/even algorithm in different languages.
- Note how more human-readable is the Python implementation compared to the other implementations.

Python

```
number = int(input('Enter a number: '))
if (number % 2) == 0:
        print('%s is Even' %number)
else:
        print('%s is Odd' %number)
```

C++

```
#include <iostream>
using namespace std;

int main() {
  int n;

  cout << "Enter an integer: ";
  cin >> n;

  if (n % 2 == 0) {
    cout << n << " is even.";
  } else {
    cout << n << " is odd.";
  }

  return 0;
}
```

C++ code source

Assembly

```
org 100h
.model small
.data
 msg1 db 10,13,"Enter Number:$"
 msg2 db 10,13,"Number is Even..$"
 msg3 db 10,13,"Number is Odd..$"
.code
main proc
 mov ax,@data
 mov ds,ax
 lea dx,msg1
 mov ah,9
 int 21h
 mov ah,1
 int 21h
 mov bl,2
 div bl
 cmp ah,0
 je even
 lea dx,msg3
 mov ah,9
 int 21h
 mov ah,4ch
 int 21h
 even:
 lea dx,msg2
```

```
 mov ah,9
 int 21h
main endp
ret
```

Assembly code source

Some other implementations

Let us implement two flowcharts in `Python`.

Convert Fahrenheit to Celsius

```
f = float(input('Enter a temperature in Fahrenheit: '))
c = 5/9*(f-32)
print(c)
```

Find and print the even numbers between `1` and `70`.

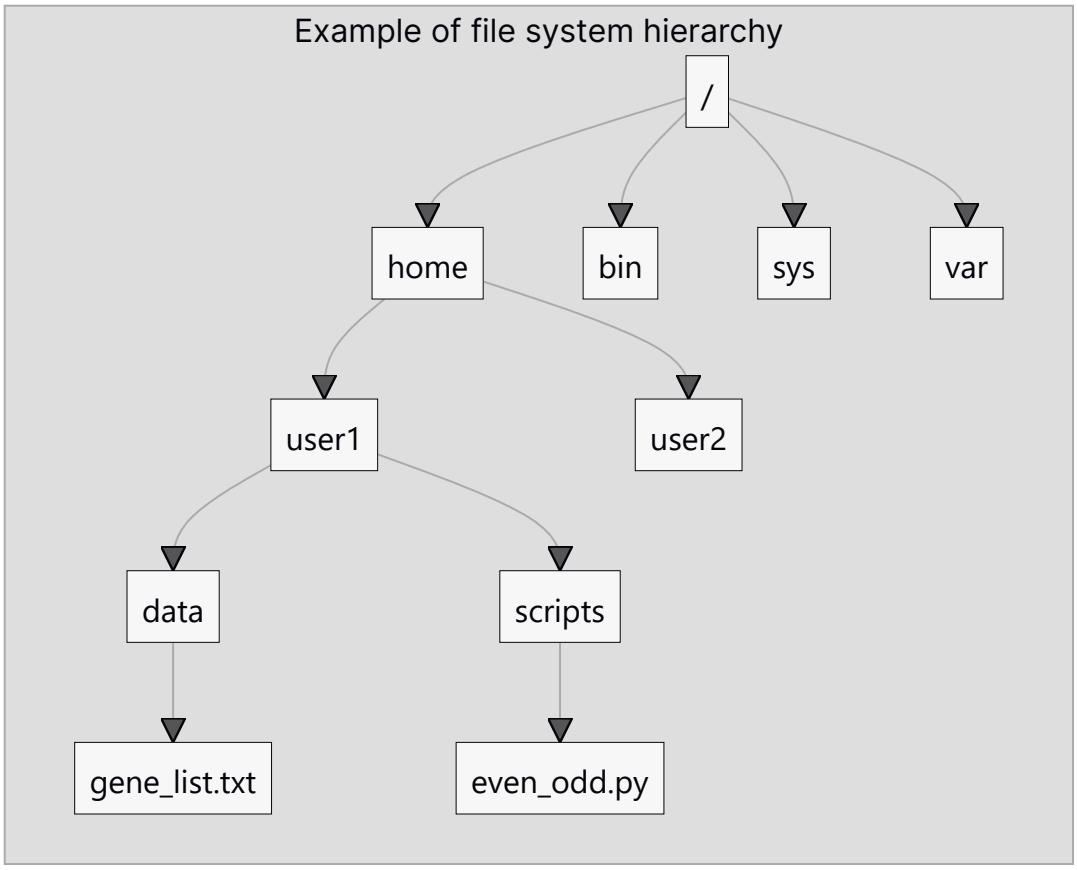
```
for counter in range(1,71):
        if counter % 2 == 0:
                print(counter)
```

Unix commands and tools

Computer clusters and **supercomputers** typically utilize a **Unix-like operating system**, such as Linux. When dealing with extensive biological data analysis, the employment of computer clusters becomes essential. Consequently, acquiring proficiency in Unix commands and tools proves beneficial.

Navigating the file system

Operating system (OS) manages computer resources such as the **computer storage**. The OS **file system** manages the computer storage by creating a **hierarchy of directories and files**. Directories may contain other directories (i.e., sub-directories) and files. Files contain **data** e.g., a file may contain a list of interesting genes.



- The uppermost directory in the hierarchy is termed the **root directory**, and is represented by a forward slash (i.e., `/`). This directory contains everything else.
- Inside the root directory, there are multiple sub-directories, one of which is the **home directory**.
- Different users of a system will have **their own directory** under the **home directory** which is called **user's home directory**.
- Upon launching a terminal, we find ourselves in the **home directory** by default. For example, if the `username` is `user1`, the corresponding home directory is `/home/user1`.
- `/home/user1` is called an **absolute path** since it shows how to reach to the `user1` directory starting from the **root directory**.
- User's home directory is also represented by the `~` character.
- Let us assume that `user1` has **two** directories under their home directory called `data` and `scripts`. There are **two** ways to point to the `data` directory
  - Via **absolute path**: `/home/user1/data`
  - Via **relative path**: `~/data`. This path is called **relative path** because it gives the path to a resource (i.e., directory or file) relative to another directory that is **not the root directory**.
- In Unix, there are multiple commands that help us navigate the file system. Here are some:
  - `pwd` command shows where we are in the file system hierarchy.
  - `ls` command lists the contents of a directory.
  - `cd` command can be used to change a directory e.g., `cd /home/user1/data`

- If we want to go one level up e.g., to `home/user1/` from `home/user1/data`, we can use `cd ..`
- `.` points to the current directory, thus if we type `cd .`, we stay in the same directory.
- if we want to go to the root directory, we can use `cd /`
- If we want to go to our user's home directory, we can use `cd ~`
- If we want to got to the previous directory, we can use `cd -`

These were a few commands to get us started.

⁂

Chapter 2. Biological data

Biological data refers to information and measurements derived from living organisms and their products. Already, there exist a diverse array of data that is vital to understanding living organisms, and ongoing technological advancements continue to expand the diversity of generated biological data. For example, **genomic** data unveils the DNA sequences that encode genetic information, while **transcriptomic** data reveals the specific RNA molecules being transcribed from those genes. **Proteomic** data offers insights into the protein composition and interactions occurring within cells or organisms. **Metabolomic** data provides a glimpse into the small molecules involved in biochemical processes. **Epigenomic** data delves into heritable changes in gene expression not caused by DNA sequence alterations. **Structural** data provides insights into the three-dimensional arrangement of biomolecules. Additionally, **phenotypic** data covers observable traits, behaviors, and physiological characteristics. These varied data types, their analysis, and interpretation are essential for achieving comprehensive insights into the intricacies of biological systems.

In this chapter, we will provide an overview of how **sequence data**, a form of **biological data**, is generated. Our **emphasis is on sequence data** because it has become a foundational component of genomics, transcriptomics, and epigenomics in unraveling the intricacies of biological systems. In genomics, sequence data reveals the precise **arrangement of nucleotides** in an organism's DNA, enabling the identification of **genes**, **regulatory regions**, and **variations**. Transcriptomics utilizes sequence data to capture RNA molecules, shedding light on gene **expression levels** and **alternative splicing patterns**. Epigenomics explores modifications that influence **gene activity** without altering the DNA sequence, with sequence data aiding in mapping **epigenetic marks** like **DNA methylation** and **histone modifications**. Collectively, sequence data empowers researchers to decipher genetic information, gene regulation, and epigenetic influences, providing a comprehensive understanding of how organisms function and develop.

We will begin by going through one of the first sequencing methods, namely Sanger sequencing, developed in 1977. Subsequently, we will explore a prevalent **high-throughput sequencing** (HTS) approach in use today, namely Illumina sequencing.

Sanger sequencing

The advent of the Sanger sequencing method used for determining the nucleotide sequences in DNA by Frederick Sanger and his colleagues in 1977 started a new era in the measurement of biological molecules and the production of biological data.

This method was an improvement in terms of simplicity and accuracy of an earlier method called “plus and minus method” created by Sanger and his colleague.

The Sanger method uses **DNA polymerase I** for the transcription of DNA under controlled conditions. DNA polymerase is an enzyme that catalyzes the synthesis of DNA molecules from nucleoside triphosphates and it is used by organisms to replicate their DNA. Additionally, this method uses 2'-deoxynucleoside triphosphates (dNTP; `2'` is read *two prime*) and 2',3'-dideoxynucleotides triphosphates (ddNTP) for the elongation and termination of nucleotide chains. ddNTPs lack a hydroxyl group on their 3' carbon of the sugar, in addition to the 2' carbon, and once incorporated in a nucleotide chain by the DNA polymerase I, the chain cannot be elongated further or in other words the chain is terminated. Phosphorus-32 isotopes (^{32}P) are used for subsequent imaging. The random elongation and termination results in DNA fragments of different size. These fragments are then fractionated by electrophoresis on acrylamide gels and followed by imaging. The pattern of bands in the image can be read to determine the DNA sequence. Since 1977 the Sanger method has undergone multiple improvements and automation. For example, the attachment of fluorescent dyes to nucleotides provided the machine-reliability capability in 1986. In 1987, the first automated DNA sequencer was developed and used to analyze and determine the structure of a gene in rats.

Note

In 1977, other sequencing methods were being developed, for example, the Maxam–Gilbert sequencing method, but the Sanger method is more popular. Can you tell why, for example, the Maxam–Gilbert sequencing did not become as popular as the Sanger method? (**hint**: read the *History* section of the Wikipedia entry on Maxam–Gilbert sequencing).

Check

Watch these 6-minute and 3-minute videos about the Sanger sequencing method. They provide different details about the same topic

Note

Even though Sanger method is more than 40 years old, it is still being used, for example, for validating results obtained from HTS approaches.

High-throughput sequencing

In the original Sanger sequencing method, we could only sequence **a few templates** simultaneously. A **template** refers to a **DNA fragment** that is going to be sequenced. Improvements to the Sanger method has increased this limit. For example, the 3730xl capillary sequencing machine from the *Applied Biosystems* can sequence **up to 96 templates** at once. All in all, it took around 25 years from the advent of the Sanger method, for the high-throughput sequencing methods to emerge, enabling the simultaneous determination of the sequence of **millions of DNA templates**.

> **Info**
>
> HTS approaches are also called **next-generation sequencing**, **deep sequencing**, and **massively parallel sequencing**.

HTS methods differ from the Sanger method in how they construct their **sequencing libraries** which improves the time required for library construction from approximately **one week** to approximately **2 days** (Mardis, ER., 2016). This and other capabilities and advances have made HTS a reality and enabled **genome-wide** (rather than site-specific) characterization of the genome, epigenome, and the transcriptome.

Currently, there are several commercially available **HTS platforms/instruments**. They are distinguished from one another depending on their **throughput**, cost, **the** typical **errors** they make, the type of output read (e.g., **single-end** or **paired-end** reads), the **read length**, etc. Each combines different methods and protocols to achieve its goal of sequencing the DNA. These methods can be classified into **four broad groups**, namely, **template preparation**, **sequencing chemistries**, **imaging/detection**, and **data analysis** (Goodwin, S. et al., 2016 and Metzker, ML., 2010).

> **Note**
>
> In this chapter and chapter 5, we focus on the Illumina sequencing technology as Illumina sequencers are currently still one of the most commonly used short-read sequencing platforms.

To prepare the template, **sample DNA** needs to be broken into **smaller fragments**. Next, DNA fragments may undergo **size selection**, where only DNA fragments within a certain range of length are retained. This is because different sequencing instruments work optimally with DNA fragments that are within a certain size range.

The next step in template preparation is either the **clonal amplification** of the templates where single DNA molecules are cloned or single DNA-molecule templates that do not rely on clonal amplification. In approaches that rely on clonal amplification, an **amplification step** is required so that there is a **strong enough signal at the imaging/detection step** required for reliable detection of the incorporated nucleotides. In the case of the clonal amplification, a set of common **adapters** are **ligated** at each end of the DNA fragments. A sequence adapter is a short chain of nucleotides that facilitates the amplification cycles as well as anchoring the ligated DNA fragment to a surface. In addition to sequence adapters, **primers** are used. Primers mark the starting point for DNA synthesis and sequencing reaction by providing a free 3' hydroxyl group to which a DNA polymerase can add a new nucleotide. Platforms using the clonal amplification, use **different strategies** to achieve amplification. For example, one strategy is to use a **solid surface** to perform the amplification as used in Illumina sequencers. Primers are bound to the solid surface (i.e., **flow cell** in Illumina) in a covalent manner and the **single-stranded DNA (ssDNA) templates** can bind to the primers by **hybridization**. **Polymerase chain reaction** (**PCR**) is mainly used for the clonal amplification step and it may introduce some challenges due to its varying **amplification efficiencies** for DNA sequences with varying proportions of G and C bases (i.e., **GC content**; Metzker, ML., 2010).

For **sequencing**, two main different methods can be enumerated i.e., **sequencing by synthesis** (SBS), **used** by the first sequencing platforms, which uses **DNA polymerase** enzyme and **sequencing by ligation** (Landegren, U., 1988) which employs **DNA ligase** enzymes for identifying the nucleotide composition of a DNA sequence.

SOLiD and **BGI** sequencers are two platforms that use **sequencing by ligation**. Illumina sequencers use SBS resembling partially to the Sanger method. During a **sequencing cycle**, which results in the identification of one of the bases of a DNA fragment, a mixture of **four different types** of **fluorophore-labeled nucleotides** that lack hydroxyl group on the 3' carbon of the sugar backbone and DNA polymerases are added to the flow cell. DNA polymerases incorporate the nucleotides in the elongating sequences. In theory, each sequence cannot be elongated more than once in each cycle due to the lack of 3' hydroxyl group. At this stage, the elongating sequences are ready to be imaged.

For **imaging/detection**, different methods include the **optical measurement** of signal intensity e.g., using **two-** or **four-color imaging**, or **non-optical measurement** of the **changes in ionic concentration**, for example in the Ion Torrent platform (Rothberg, JM. et al., 2011). Illumina sequencers use two or four laser channels (depending on the platform) to excite the fluorophores bound to incorporated nucleotides in a given sequencing cycle and use total internal reflection fluorescence microscopy to image/detect the bioluminescence from the clusters on the solid surface. Two advantages of two-color imaging over four-color imaging is its higher speed and

lower cost since less imaging is performed and less fluorophore is used. In Illumina, after the imaging in one cycle, fluorophores are cleaved and washed away from the flow cell. Additionally, the missing 3' hydroxyl groups are regenerated so that in the next cycle, the elongation can continue.

HTS methods are high-throughput since they can perform sequencing followed by detection (e.g., via imaging) parallelly for **millions of DNA fragments/templates** as they **do not rely on the electrophoresis step** that is required in the Sanger method. This is possible, using Illumina technology as example, because after the clonal amplification of DNA fragments, each colony occupies distinct sites and additionally, they can undergo sequencing reactions in a parallel fashion.

Further advances in NGS technologies such as **improved sequencing chemistries** as well as increased **detection sensitivities** have resulted in higher throughput i.e., higher amounts of data being produced per sequencing run.

The ultimate output of HTS technologies is a large amount of **sequencing reads** where each read determines the sequence of bases of a single molecule of DNA that underwent sequencing. If a DNA fragment is sequenced in one direction, this results in **single-end reads**. However, if a DNA fragment is sequenced in **both forward and reverse directions** (i.e., from each end), it results in **paired-end reads**. In general, the sequence read length generated by the commonly used HTS methods is **between 30 and 400 nucleotides long** which is **shorter than 600 to 800 base pairs of length achieved by the Sanger method**. Short sequencing reads introduce a few challenges, for example, during genome assembly or alignment and downstream analyses (see **chapter 4**). Other HTS technologies with longer read length address some of these challenges, however, they are currently **more expensive** and suffer from **lower throughput** resulting in the popularity of cheaper platforms with shorter sequence read length.

✓ **Check**

Watch 4 minutes and 23 seconds of this video about the Illumina **sequencing by synthesis**. Do not worry if this video feels a bit complicated at first. We will learn more about Illumina later.

Note

This video illustrates **paired-end sequencing**. Later, we learn about the benefits of paired-end sequencing over **single-end sequencing**, when we are discussing sequence alignment.

⁂

Chapter 3. Biological databases

There are thousands of biological databases available with the aim of storing biological data. Annually, the *Nucleic Acids Research (NAR)* journal issue on biological databases compiles a list of such databases, offering insights and advancements in the realm of storing biological data. The sheer volume of these databases has become so extensive that a database such as the NAR online Molecular Biology Database Collection has been established to systematically monitor and categorize these databases.

These biological databases offer data about different topics/classes such as:

- Publicly available nucleotide sequence data
- Publicly available protein sequence data
- Functional information about genes and gene products
- Gene structure, regulatory elements, variants
- Proteins, their families, domains, conserved sites
- Biological pathways
- Biomedical literature
- and many more...

Question

Some classify biological databases into **three** categories: **sequence**, **structure**, and **function**. Can we say that there can be as many classes of databases as there are biological data types? Why? Why not?

Here, we learn more about the **biological databases** as a **medium for storing and retrieving biological data**. We get to learn about a few biological databases.

"Evolution" of biological databases

Before we learn about some modern databases, let us first look back in time for a while. The **first bioinformatics database** was published in **1965** as a **printed book** by **Margaret Dayhoff** and her colleagues under the name ***Atlas of Protein Sequence and Structure*** and it contained **65 protein sequences**. A lot has changed since 1965!

Let us check some of these changes with a **focus on DNA sequences** but we can assume that similar trends can be traced with regards to other biological data as well.

- The first **DNA sequencing methods** emerged around **1977** and in the **same year** the **first full DNA genome of bacteriophage φX174** was sequenced. Such sequence data were stored in different sequence databases.
- In **1980**, the **world's first nucleotide sequence database**, the **European Molecular Biology Laboratory (EMBL) Nucleotide Sequence Data Library** was established with the **goal of creating a central database of DNA sequences**, rather than have scientists **submit sequences to journals**.
- In **mid 1980s** (i.e., 1986 and 1987) some sequence databases such as the **EMBL's Nucleotide Sequence Data Library**, **NCBI's GenBank** and **DNA Data Bank of Japan (DDBJ)** formed a **union** and agreed upon some **standards** in order to **make it easier to share data among different databases**.
- Advances in methods and technologies sped up the data generation. For example, the **first automated DNA sequencer** was developed in **1987** by Lloyd M. Smith, working with Applied Biosystems.
- The **Human Genome Project (HGP)** launched in **1990** with the goal of sequencing the human genome.
- The emergence of the **World Wide Web** in the **1990s** helped such **databases** to become **available online**.
    - The **EMBL Nucleotide Sequence Data Library** became accessible on the Web in **1993**.
    - The **NCBI website** was made available online in **1994**.
- In **1996**, the Bermuda Principles as a result of the Bermuda accord came to being and it required that all DNA sequence data produced by the HGP to be released in **publicly accessible databases** within **24 hours after generation**.
- Since 2008 and with the emergence of **next-generation sequencing methods**, a wealth of sequence data has been generated and gathered in online databases.

So, we can see that biological databases started as **printed books**, and evolved overtime to be distributed via **magnetic tapes** (e.g., protein data bank (PDB)) and **DVDs** and now they are accessible via **the Web**. The **content** also have grown from **tens** to **millions** of entries.

Finding relevant biological databases

For our studies, work, and research, we may benefit from using biological databases. For example, we can use data from available databases to **compare** our **experiment results** with **what is already known** or we can use external data to **annotate** our results.

Example

Imagine that we have obtained genomic DNA from the blood of a cohort of people and performed DNA sequencing. After *genome variation** analysis, we have found a change in a particular position of the genome and we want to know how common this change is in different human populations. We can use, for example, gnomAD for this purpose.

* We will learn about genome variation later.

Example

You may want to get the genomic coordinate of all genomic loci that are known to be bound by a specific transcription factor. We can use, for example, Gene Transcription Regulation Database (GTRD) or UniBind for this purpose.

Question

How can we find a relevant biological database?

Answer

Here are some possible ways:

- Browsing through the NAR online Molecular Biology Database Collection
- Following NAR journal issues on biological databases
- When reading scientific articles, we can look for mentions to relevant biological databases, for example, in their methods sections
- Recommendations from our supervisors, colleagues and collaborators
- Can you suggest more?

Learning how to use a biological database

After having found a biological database, the next step is to learn to use it efficiently.

- Using many of the biological databases are rather intuitive.
  - There is almost always, a **search box** and a **search button**.
- However, **it may require a bit of time and effort to learn how to use a database properly** but overtime you, become very good at it.

> **Tip**
>
> One of the best places to start is the **biological database website itself** and looking for pages such as **About**, **Help**, **Frequently asked questions** (or FAQs), **How to**, **Getting started**, **Guide**, **Tutorial**, **Wiki**, **Overview**, **fact sheet**, **cheat sheet**, **documentation**, **Introduction** etc.
>
> - Note that biological databases are often updated with new features or the layout may get updated. If there is a change, there is a better chance that these pages get updated faster than other external resources and tutorials.
>
> Often it pays off to spend a bit of time reading through these pages (especially going through the **FAQ**s if there is one) and familiarize yourself with the database even if you are in a hurry.

> **Example**
>
> NCBI has created **fact sheets**, **How to**'s, etc. about their resources which are good resources for getting started with a specific resource or tool. The list can be accessed here (**note**: scroll down the page to see the fact sheets and How to's). For example:
>
> - NCBI Gene
> - NCBI RefSeq
> - NCBI ClinVar
> - NCBI RefSeqGene

Concepts and terms

Here are some useful concepts and terms related to databases in general. You may encounter these concepts, for example, when you are reading the guides, FAQs, and

tutorials about a database. Thus, it is a good idea to know them.

- **Database record and field**. A **record** is a **collection of information** about an **entity** or **object** (e.g., *TP53* gene). This information may be organized in different **fields**. For example, in the case of *TP53* gene, one field may store the *gene length*, the other may store the **start** and **end coordinates** in the genome, etc.
- **Database query**. A request to access data from a database.
- **Keyword** or **query term**. The word(s) that one uses when querying a database.
- **Fielded keyword**. It tells the database to search the keyword in a specific **field**. Searching a field is known as **fielded search**.
- **Filter**. Filters can be used to restrict a query results.
- **Stable identifier**. An identifier used by a database to refer to a record, for example, a feature, such as a gene, a transcript, an exon or a protein. In the case of biological databases, an identifier is stable because it does not change after updates to the database and continue to refer to the same genomic feature. This could not be achieved if we have used a gene name to refer to a genomic feature because there is a chance that the name would change in the future (e.g., `ENSG00000169083` is an Ensembl gene identifier).
  - **Accession number**. A **unique identifier** assigned to a biological database **record** e.g., in a DNA sequence database. Accession numbers **do not change over time**. Accession numbers can be used to **connect/link data from two or more databases together**.
- **Cross-reference**. When one record in one biological database is linked to another record in another biological database via an accession number.

> **Note**
>
> Certain databases are vast in scope, making it impractical to grasp their entirety in one go. Rather than expecting immediate mastery, it is better to learn about each database gradually.
>
> When working with databases, you may encounter concepts that are unfamiliar to you. Some of these concepts will be covered later here. So, do not worry if you do not understand them now!

⁂

Chapter 4. High-throughput sequencing

In chapter two, we briefly learned about two methods to sequence DNA, namely, the Sanger sequencing and the high-throughput sequencing (HTS). In this chapter, we look a bit more into the HTS method focusing on the Illumina technology since Illumina sequencers are currently commonly used.

To see the benefits of the HTS technology, we can compare it to the Human Genome Project (HGP) whose goal was to sequence the human genome:

- Using an HTS technology (e.g., a state-of-the-art Illumina sequencer), one can sequence the human genome in **a single run** on **a single instrument** in **a few days**. HGP took around `13` years to complete!
- State-of-the art sequencers enable whole genome sequencing for less than US$ `1000`. HGP costed US$ $3 * 10^9$.

> **Note**
>
> - HTS is also referred to as: **deep sequencing**, **next-generation sequencing (NGS)**, and **massively parallel sequencing**.
> - Here, we focus on **bulk data** (i.e., nucleic acid material is coming from more than one cell, sometimes thousands of cells.) and not single-cell data.
> - Here, we use the terms **sequencing instrument**, **sequencing machine**, and **sequencer** interchangeably.
> - We also briefly learn about the HGP in a future chapter.

Illumina high-throughput sequencing steps

We can enumerate **four** primary steps in high-throughput sequencing using Illumina technology. Some of these steps are shared with other sequencing technologies. Here, we briefly go through them.

1 Nucleic acid isolation/extraction

There are different protocols (and kits) for isolating and extracting the nucleic acid material of interest, for example, extracting

- Genomic DNA (gDNA 🧬) from tissue, blood, cell, ...
- Total-RNA (i.e., all RNA species in a sample), or only mRNA, ...
- ...

2 Sequencing library preparation

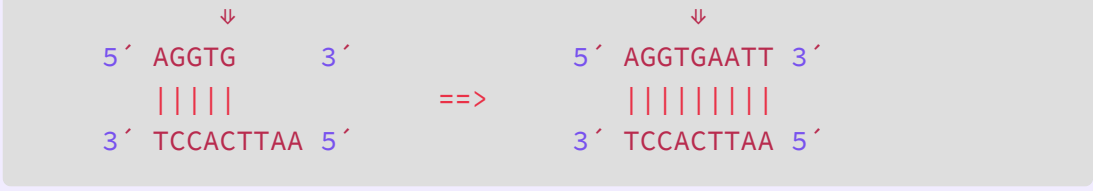
Just as a physical library houses thousands of books, a **sequencing library** contains hundreds of thousands or even millions of nucleic acid molecules (such as DNA or RNA) that are destined for sequencing.

By **library preparation**, we mean modifying the nucleic acid molecules in such a way that they are suitable for a particular sequencing instrument. This involves several steps, some of which are:

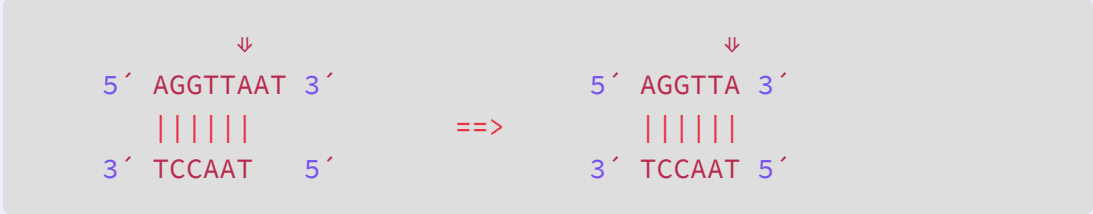
- **Nucleic acid fragmentation** (or **shearing**)
  - Sequencing instruments have a preference in terms of the **nucleic acid fragment size** that they can sequence efficiently.
  - For Illumina, the typical fragment size is ~350-500 base pairs depending on the application.
  - There are different ways for fragmentation
    - **Mechanical** e.g., sonication
    - **Enzymatic** e.g., using restriction enzyme(s)
- **End repair**
  - The fragmentation step may result in `3´` and `5´` overhangs.
  - These need to be repaired before adding the adapters (see. next step).

    Example

    ```
    End repair of 5´ overhang
    ```


    ```
    End repair of 3´ overhang
    ```

- **Sequencing adapter ligation**

    - **Adapters** are a short sequence of DNA or RNA (oligonucleotides) attached to each end of a fragment.
    - Adapters, for example, in Illumina sequencers, help binding of the fragments to the flow cell.
    - Illumina adapters may include **indices** that can be used for *sample multiplexing* allowing sequencing of many samples in one sequencing run.
    - **Note** that different sequencing instruments have their own set of adapters.
- **Size selection**
    - Select fragments with size within a certain range expected by a sequencer.
    - Remove other byproducts such as adapter dimers.
- **Library amplification**
    - This step might be needed if the there was not much of starting material.
    - This can be achieved through methods such as polymerase chain reaction (PCR), referred to as a **PCR-based method**. However, because PCR can introduce biases, alternative workflows exist that avoid PCR as much as possible and are termed **PCR-free method**.
    - **Note**. This step is separate from the **cluster generation** in the sequencing step (we learned about the cluster generation in the video about Illumina sequencing in **chapter 3**).
- **Quality control**
    - Checking whether the library preparation has been successful.
    - Checking whether the molecules that are going to be sequenced have an optimal concentration.
- **Target enrichment**
    - This step is needed when our focus is on specific genomic regions, such as exclusively protein-coding regions, as seen in the case of whole exome sequencing (WES).

3 Sequencing

During this step, the fragments that were prepared in the sequencing library preparation step undergo the actual sequencing process. We learned about the sequencing step in **chapter 3**.

4 Bioinformatics data analysis

After the sequencing step, the subsequent step involves the **bioinformatics data analysis**.

Part of the data analysis (called **primary analysis** in the case of Illumina), is done on the Illumina sequencing machine itself. For example:

- The Illumina instrument uses its **built-in software** called **Real-Time Analysis (RTA)** and analyzes the images from the sequencing step which results in **base calls** and assignment of **base calling quality score** to each of the base calls.
- If many samples were pooled/multiplexed in a single sequencing run, they are **demultiplexed** i.e., the sequencing reads are assigned to different samples using the indices in the adapters.

HTS instruments such as Illumina produce **millions of reads**. The raw sequencing data is typically stored in **FASTQ files**. Here, we see a toy example of how a DNA fragment is sequenced from its both ends and the sequencing reads are stored into `2` FASTQ files.

Example

Paired-end sequencing of a **DNA fragment**.

```
   ADAPTER------read1------>                                ADAPTER
5´ NNNNNNNAGGTTAATGCTTTCTCTCTA.........CACTGAGATCTTAGGAGAAANNNNNNN 3´
       |||||||||||||||||||||||||||||||||||||||||||||||||||||||||
3´ NNNNNNNTCCAATTACGAAAGAGAGAT.........GTGACTCTAGAATCCTCTTTNNNNNNN 5´
                                    <------read2-------
```

In paired-end sequencing, a DNA fragment is sequenced from its both ends resulting in two sequencing reads: *read1* and *read2*. These are stored in two FASTQ files.

Note that the DNA fragment to be sequenced is also referred to as an *insert*, since the DNA fragment is inserted between sequencing adapters.

Remember that an Illumina sequencer sequences each of the strands separately and one after another. The illustration above is only meant to show from where each of the reads is originating.

Below, we can see the content of two FASTQ files with reads from the DNA fragment above.

This is *read1* obtained from the DNA fragment above (stored in the 1st FASTQ file).

```
@HWI-ST898:563:C4CK5ACXX:5:1212:14521:85648/1
AGGTTAATGCTTTCTCTCTA.....
+
=?>=@AAADCC@@BBBBBB@.....
```

This is *read2* obtained from the same DNA fragment (stored in the 2nd FASTQ file).

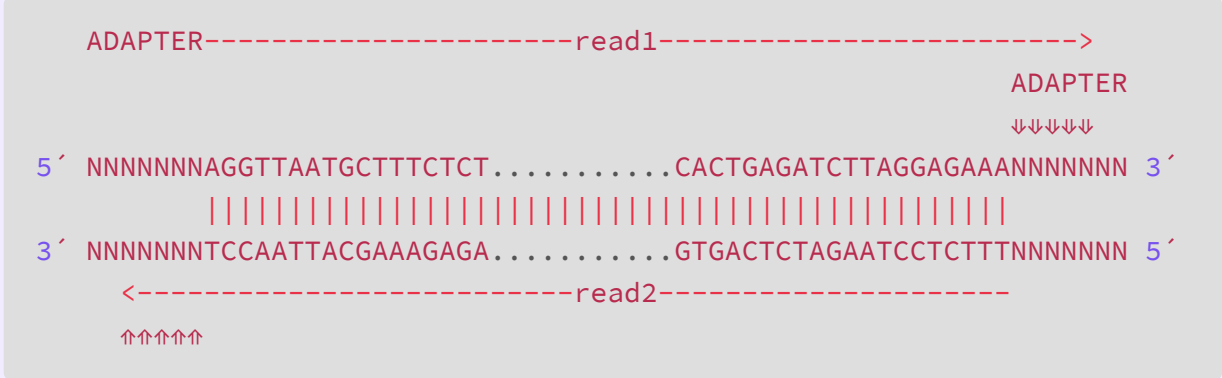
```
@HWI-ST898:563:C4CK5ACXX:5:1212:14521:85648/2
TTTCTCCTAAGATCTCAGTG.....
+
>><@BBBC@@C@@BBBBC?C.....
```

Once we have a set of FASTQ files, usually the next step is to use a bioinformatics tools such as **FastQC** to perform **quality control**. This step tells us if we need to take further actions before proceeding with downstream analyses. Here are two examples:

- We may have noticed that the closer we get to the end of a sequencing read, the more quality score drops. Sometimes the quality of these bases are below a certain accepted threshold. In such a case, we need to **trim** these bases using a bioinformatics tool such as **Trimmomatic**.
- Let us assume that we have decided to do paired-end sequencing and that each read is `75` nucleotides long (i.e., we are going to do `75` sequencing cycles for each read). If the DNA fragment is < `150` bp, then *read1* and *read2* will **overlap** and we end up sequencing a portion of the DNA fragment twice. In this example, if the DNA fragment is smaller than `75`, *read1* ends up having the adapter sequence that was attached/ligated to the other side of the fragment. This is known as *adapter read-through*. FASTQC or other similar bioinformatics tools can tell us if we have adapter sequences in our data. In case we detect adapter sequences in our data, we can use, for example, Trimmomatic to remove the adapter sequences before we continue with the downstream analyses.

Example

Adapter read-through



Sequence alignment

We learned that to sequence the DNA, we need to break it down into smaller fragments, inevitably leading to the loss of location information within the genome. The primary objective of sequence alignment is to precisely determine the genomic positions of the sequencing reads derived from these fragmented DNA segments. It is important to note that, in this context, we operate under the assumption that we possess prior knowledge of the genome's sequence, meaning **we have access to a reference genome for comparison**.

Illumina produces a large number (e.g., tens of millions) of **short reads** that needs to be **aligned** to a **reference genome**. This is a **computational challenge** because we need to align the sequencing reads in a reasonable time (and also make sure that we do not **run out of memory**). We also need to take into consideration that the **sequencing** step is **not perfect** and if, for example, base calling errors occur during the sequencing step, finding the genomic location of a sequencing read with error(s) becomes challenging or sometimes impossible.

Many commonly used alignment algorithms use a **seed-and-extend** approach to perform **short read alignment** to a **reference genome**. In this approach, we start by taking a sub-sequence (i.e., the seed) of the read we want to align and try to find a **match** between this sub-sequence and part of our reference. Once, we find one or more matches, we take each and try to **extend** the sub-sequence and check whether we get a better match between the extended sub-sequence and the reference. The extension step uses a technique called **dynamic programming** (we will learn about sequence alignment using dynamic programming in **chapter 6**).

At best, we find only one position in the reference genome that a read matches and extends properly. It is possible, however, that a read aligns to more than one position. This type of read is known as a **multi-mapped read**. There is also a chance that a read cannot be aligned to the reference genome. This type of read is known as **unmapped read**. This can be, for example, due to sequencing errors, reading coming from repetitive areas in the genome, or it is possible that the read is coming from a contaminating organism.

Example

Sequence alignment.
In this example, we have a reference genome, and a set of sequencing reads. We want to align these reads to the genomic region they have originated from.

```
CTAGAGCGTGGCCCGGAGCTGCCCTTTCCTCTTCGGTGAAGTTTTTAAAAGCTGCTGCGA #
reference

CCGGAGCTGCCCTTTCCTCTTCGGTGA
CCTTACCTCTTCGGTGAAGTTTTTAAA
GAGCTGCCCTTTCCTCTTCGGTGAAGT
TGCCCTTACCTCTTCGGTGAAGTTTTT
CCCGGAGCTGCCCTTTCCTCTTCGGTG
TACCTCTTCGGTGAAGTTTTTAAAAGC
```

After alignment.

```
                          ⇓
00000000011111111112222222223333333333444444444455555555556 #10s
12345678901234567890123456789012345678901234567890123456789 #1s
CTAGAGCGTGGCCCGGAGCTGCCCTTTCCTCTTCGGTGAAGTTTTTAAAAGCTGCTGCGA #Ref
            CCGGAGCTGCCCTTTCCTCTTCGGTGA
                      CCTTACCTCTTCGGTGAAGTTTTTAAA
               GAGCTGCCCTTTCCTCTTCGGTGAAGT
                   TGCCCTTACCTCTTCGGTGAAGTTTTT
          CCCGGAGCTGCCCTTTCCTCTTCGGTG
                                    TACCTCTTCGGTGAAGTTTTTAAAAGC
                          ⇑
```

🧩 In a real situation, the **reference is billions of nucleotides long** and we have **hundreds of millions of sequencing reads** each are **e.g., 100 nucleotides long**.

❓ Incidentally, what is happening at the position `27` marked by the arrow?

Once the reads are aligned to a reference genome, the results are stored in a standard file format called Sequence Alignment/Map (SAM). This is a **text file format** and thus it is human-readable but it **takes lots of space**. The binary alignment map (**BAM**) is a **binary file** that stores similar information in a **compressed form** thus uses less space but it is **not human-readable**. In addition to where in the reference genome a read is aligned, the aligner tool stores other information in the SAM/BAM file. For example, there is a `FLAG` column that can tell whether a read was mapped, unmapped, etc. There is another column called `MAPQ` (mapping quality) that reports a

score, once converted, it shows what is the probability that a read was aligned to a wrong position.

There are bioinformatics tools such as samtools that can be used to work with SAM/BAM files.

Breadth and depth of coverage

Two concepts relevant in the context of HTS*, *are the* (Breadth of) coverage *and the* depth of coverage*.

Definition

(Breadth of) coverage. The percentage of a genome that has been sequenced at least once in a sequencing experiment. In other words, the proportion of the genome covered with at least one read.

- **Example**. A `95`% breadth of coverage means that we have successfully sequenced `95`% of the genome at least once.
- A high breadth of coverage implies that a large portion of the genome has been sequenced, providing a more complete picture of the genetic information.

Depth of coverage (aka. sequencing depth). The average number of reads across the genome. In other words, the average number of times a particular base in the genome has been sequenced.

- **Example**. A depth of coverage of 30X means that, on average, each base has been sequenced 30 times.
- A higher depth of coverage increases the accuracy and confidence in the sequence data. Certain applications such as variant calling require a high depth of coverage.

Note

Depth of coverage can be calculated/estimated in two ways:

Raw Read Depth (of coverage). The total amount of sequence data produced by a sequencer (before alignment), divided by the reference genome size.
$C = LN/G$
where C is depth of coverage; L is the read length; N is the number of reads; and G is the genome length.

Mean mapped read depth (of coverage). The sum of the aligned read depths at each reference base position, divided by the reference genome size.

Example

Let us consider a `30`-nucleotide long genome and `12` reads each `5` nucleotides long. The breadth and depth of coverage is calculated for a few different situations (assuming all reads can be aligned to the reference genome).

```
# Breadth of coverage: 100%. Depth of coverage: 2X.
     _____     _____     _____
_____     _____     _____
     _____     _____     _____
_____     _____     _____
============================== # refernece
000000000111111111122222222223 # position-10s
123456789012345678901234567890 # position-1s

# Breadth of coverage: 50%. Depth of coverage: 2X.
     _____
_____     _____
     _____
_____     _____
     _____
_____     _____
     _____
_____     _____
============================== # refernece
000000000111111111122222222223 # position-10s
123456789012345678901234567890 # position-1s

# Breadth of coverage: 73%. Depth of coverage: 2X.
_____        _____
_____           _____
_____       _____
_____      _____ _____
_____     _____ _____ _____
============================== # refernece
000000000111111111122222222223 # position-10s
123456789012345678901234567890 # position-1s
```

Why is there a need to sequence approximately $120_{10}0^9$ bases when theoretically, sequencing around $(3_{10}0^9)_30 = 90_{10}0^9$ bases should suffice to attain a depth of coverage of `30x`?

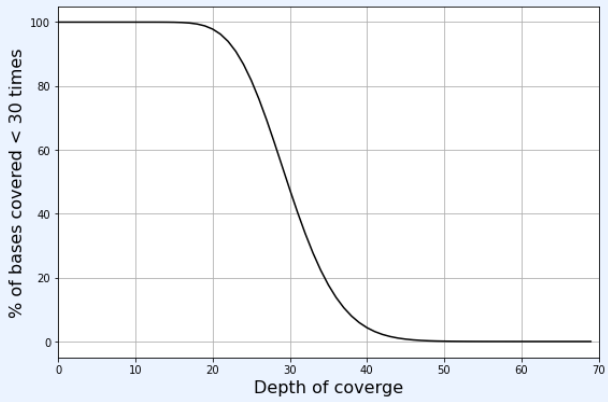
Note

In the case of the human genome, to get a `30x` depth of coverage, we need to sequence around $120 * 10^9$ bases.

❓ Why is there a need to sequence approximately $120 * 10^9$ bases, when theoretically, sequencing around $(3 * 10^9) * 30 = 90 * 10^9$ should suffice to achieve a depth of coverage of `30x`?

💡 This is because reads are **not distributed evenly across the genome**. According to Lander and Waterman (1988), these reads roughly follow a **Poisson distribution**.

The following figure uses Poisson cumulative distribution function to show that if we aim for `30x` depth of coverage, around half of the bases (i.e., `47.6`%) are covered by `29` or less reads. If we aim for `50x` depth of coverage, practically all bases are covered `30` times or more (see the *Appendix* section for the Python code used to make this figure).



Different applications require/recommend different depths and/or breadths of coverage (e.g., check this page made by Illumina giving coverage depth recommendations). Illumina has also made a sequencing coverage calculator that can calculate the reagents and sequencing runs needed to achieve the desired sequencing coverage for a particular experiment. If you are interested, you can also read this Illumina technical note on estimating sequencing coverage.

Once we have the SAM/BAM file(s), we can proceed to the downstream analyses. For example, if we have a BAM file from a whole genome sequencing, we could use it

together with a bioinformatics tool such as Genome Analysis Toolkit (GATK) or VarScan to detect variants i.e., genomic regions in our sample that are different from the reference genome that was used for the alignment (we will learn about genome variation in a chapter).

Tip

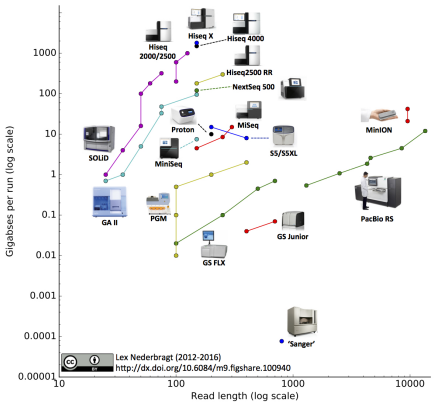
One **biological database** that we can use to access publicly available high **throughput sequencing data**, is NCBI Sequence Read Archive (SRA).

Quote

Sequence Read Archive (SRA) data, ... is the largest publicly available repository of high throughput sequencing data. ... SRA stores raw sequencing data and alignment information to enhance reproducibility and facilitate new discoveries through data analysis. source

Other high-throughput sequencing technologies

We may have encountered the fact that there are a various HTS technologies available. The figure below illustrates the relationship among some of these available sequencing machines based on their read length and throughput.



More-info

To learn more about some of these technologies, please refer to

- *Coming of age: ten years of next-generation sequencing technologies* (Goodwin et. al., 2016)

- *Sequencing technologies - the next generation* (Metzker, 2010).

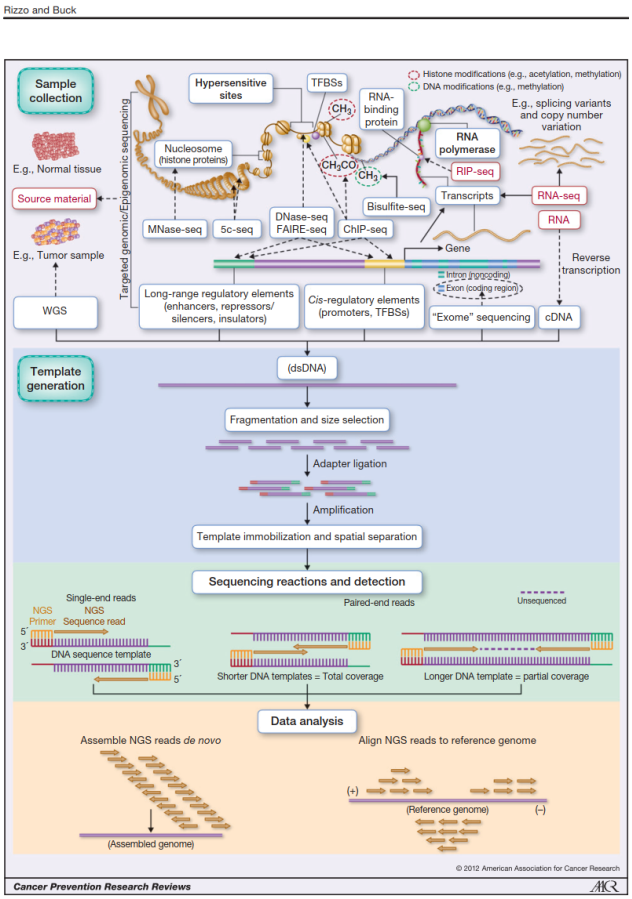
Applications of HTS data

Here, we briefly mention a few of the **applications** of the **HTS data** obtained from different assays some of which we are going to learn more about in the future chapters. **For example**, we could use

- Whole genome sequencing (WGS, sometimes referred to as DNA-seq) or whole exome sequencing (WES) data to
  - **detect variation** in the genome such as single nucleotide variants, insertions, deletion.
    - **Note**. If WES is used, we only detect the changes in the protein-coding regions of the genome.
- RNA-seq data to
  - find different **transcripts** in a cell or cells.
  - quantify the **abundance** of each detected **transcript** (i.e., to quantify the **gene expression**).
- ChIP-seq data to
  - find about genomic regions bound by DNA-associated proteins such as **transcription factors** (i.e., protein-DNA interactions)
  - find about histone tail modifications
    - For example, the acetylation of the lysine residue at N-terminal position `27` of the histone H3 protein (H3K27ac) associated with the higher activation of transcription.
- MeDIP-seq or Bisulfite-seq data to
  - determine the **patterns of DNA methylation**
- ATAC-seq, FAIRE-seq, or DNase-seq data to
  - identify **accessible regions** in the genome
- Chromosome conformation capture data such as Hi-C data to
  - identify the **3-dimensional structure and organization of the chromatin**
- A multitude of sequencing assays are continuously emerging, each tailored to diverse applications.
  - For example, check this poster by Illumina from ~2016 that visually represents tens of available sequencing assays.

The following figure from Rizzo and Buck (2012), represents the genome, epigenome (and partially the transcriptome) together with some of the available HTS assays that can be used to study it.

- **Note** that the audience of this figure are clinical researchers and physician scientists but we also can benefit from it.



Task

Check the example boxes as well as the figures in this chapter and answer the following questions.

❓ In the figure with different sequencing machines, what sequencers generate longer reads than the Sanger sequencer?

❓ According to the figure in Rizzo and Buck (2012), which regions are sequenced by exome sequencing?

❓ Which of the top or bottom strand is read by the DNA polymerase to make read1 (**tip**: take into account in which direction DNA polymerase moves, during its polymerase activity)?

```
   ADAPTER------read1------>                              ADAPTER
5´ NNNNNNNAGGTTAATGCTTTCTCTCTA.........CACTGAGATCTTAGGAGAAANNNNNNN 3´
        ||||||||||||||||||||||||||||||||||||||||||||||||||||||||
3´ NNNNNNNTCCAATTACGAAAGAGAGAT.........GTGACTCTAGAATCCTCTTTNNNNNNN 5´
                                        <------read2-------
```

Assume we have a `60` nucleotides long genome and `6` reads each `27` nucleotides long (see below) and answer the following questions.

❓ Calculate the (breadth of) coverage for the following example.

❓ Calculate the depth of coverage for the following example.

```
000000000111111111122222222223333333333444444444555555555 # position-10s
123456789012345678901234567890123456789012345678901234567890 # position-1s
CTAGAGCGTGGCCCGGAGCTGCCCTTTCCTCTTCGGTGAAGTTTTTAAAAGCTGCTGCGA # reference
            CCGGAGCTGCCCTTTCCTCTTCGGTGA
                      CCTTACCTCTTCGGTGAAGTTTTTAAA
                GAGCTGCCCTTTCCTCTTCGGTGAAGT
                    TGCCCTTACCTCTTCGGTGAAGTTTTT
           CCCGGAGCTGCCCTTTCCTCTTCGGTG
                           TACCTCTTCGGTGAAGTTTTTAAAAGC
000000000001222333344455566666666666665444333322211100000000 # pileup count
```

Further reading

- 15-minute course by Illumina telling about sequencing via Illumina technology
- This document made by ThermoFisher Scientific explains DNA sequencing library preparation for Illumina instruments.
  - **Note**: This is more up-to-date compared to the following resource. Also, it has information that we did not cover in this chapter but some of us might find it useful.
  - **Note**: Please be aware that in this document, they also promote their products.
- A set of videos (a 22-minute video and a 40-minute video) telling about library and sample preparation for Illumina Genome Analyzer offered by Broad Institute and in cooperation with Illumina.
  - **Note**: These videos date back to **2010** but they still might have interesting information.
  - The slides for these videos can be found here.

Appendix

Python code for depth of coverage figure.

```
import numpy as np
from scipy.stats import poisson
import matplotlib.pyplot as plt
%matplotlib inline # use this if you are using Jupyter notebook

depths = np.arange(0, 70)
plt.figure(figsize=(9,6))
plt.plot(depths, poisson.cdf(29, depths)*100, 'k')
plt.xlabel('Depth of coverge', fontsize=16)
plt.ylabel('% of bases covered < 30 times', fontsize=16)
plt.xlim(0, 70)
plt.grid();
```

⁂

Chapter 5. Genome and genome variation

In chapter 3, we learned that ClinVar is a database that aggregates information about **genomic variation** and its relationship to human health. we noted that some **changes/variation** in the **genome** were associated with a **particular disease** and that some of the changes were **benign**. These observations have prepared us for this chapter's topic, where we turn our focus on learning about **genome** and **genome variation**. However, we start by learning a few facts about the **human genome** and the **Human Genome Project (HGP)**.

Question

Why should we learn about or study the **variations in the genome**?

Answer

A possible answer. Studying **genetic variation** is interesting because:

- Variation provides a way for organisms to **evolve**.
- Certain **diseases** can be **explained** by **variations in the genome**. For example, a **single nucleotide change** (GAG → GTG) at a particular position in the **β-globin** gene results in a **single amino acid change** which in turn results in the **sickle cell disease/anemia** in an individual (provided that both copies of the β-globin gene in that individual has this change).
- Genomic alterations have been identified in numerous cancer types.

Human genome and the Human Genome Project

Earlier, we learned that the word *genome* refers to the **entirety of an organism's genetic information**. In human, this amounts to $3.1 * 10^9$ nucleotides (in the haploid form). Thanks to the HGP and other efforts, now, we know almost all of this sequence.

A bit of history about the HGP:

- In **October 1990**, an international collaboration, the HGP, started the **sequencing** of the **human genome**.
- In **2001**, they published the **human genome draft** that covered about **94%** of the human genome and it was around $3 * 10^9$ nucleotides long.

- The HGP project was officially **completed** in **April 2003** with the release of the **finished human genome**, but the effort to sequence the entire human genome continued.

> **Note**
>
> **The difference between the *draft* and *finished* human genome sequence**
>
> According to the NIH Human Genome Project FAQ, the difference is in the **(breadth of) coverage**, the **number of gaps** and the **error rate**.
>
> - **(Breadth of) coverage** refers to how large part of the human genome was sequenced. In **2000**, when the **rough draft** was published, only around **90%** of the human genome was sequenced. In **2001**, it was about **94%**, and in **2003**, the coverage reached **99%**.
> - The **number of gaps** changed from around **150,000** in **2000** to less than **400** in **2003**.
> - The **error rate** changed from **1 in 1000** in the year **2000** to **1 in 10,000** in the year **2003**.

- The **gaps** in the human genome are due to a few issues. For example, there are **difficult-to-sequence regions** in the human genome such as the **repetitive regions**. In addition, **sequencing errors** and **low sequence read coverage** at certain loci make it **difficult to reliably reconstruct the genome** and thus resulting in gaps. There has been an ongoing effort to **fill in the remaining gaps** in the human genome. For example, we can mention the efforts by the Genome Reference Consortium (GRC), and recently by the Telomere to Telomere (T2T) consortium.
- There are also **other efforts**, with regard to the human genome, for example, one of the aims the Human Pangenome Reference Consortium (HPRC) is to "*better represent the human diversity*" in the human genome reference and "*to address the racial and ethnic biases in genomic resources*" (e.g., check this).
- The human reference genome is a mosaic constructed from the genomes of **a few female and male volunteers** from **Buffalo, New York**. DNA material, extracted from their blood samples, was utilized for sequencing.
  - Watch this 1-minute video, where Prof. Eric Lander, the first author of the HGP project publication, tells from where the individual recruited to the HGP.

More-info

In case you are interested to find out more about the **DNA donors** in the HGP, you can read the following:

- The NIH Human Genome Project FAQ and the answer to the question *Whose DNA was sequenced?*
- Genome reference consortium FAQ and the answer to the question *How many individuals were sequenced for the human reference genome assembly?*
- Genome donors section of the Wikipedia entry on the HGP.

Info

Browsing the *NCBI Gene* database, we encounter terms such as `GRCh38`, `GRCh37`, and `T2T-CHM13`. We also might have already encountered `hg18`, `hg19` and `hg38`. What do these refer to?

These refer to **different versions of the human genome reference**. For example, `hg18` (aka `NCBI36`) was released in `2006`. `hg19` (aka `NCBI37` or `GRCh37`) was released in `2009`. `hg38` (aka `GRCh38`), was released in `2013`. The most recent version of the genome reference is the `T2T-CHM13` from the Telomere-to-Telomere consortium, released in 2022. However, the `GRCh38` reference genome continues to be widely used.

As we have learned so far, the **latest versions** tend to **fill in the remaining gaps** in earlier versions as well as adding more capabilities and fixing possible issues. For example, `hg19` has **alternate haplotype** assemblies for a few chromosomes, something that was not in `hg18`.

Note: These (i.e., `GRCh38`, `GRCh37`, etc.) are also called **major release**.

Note: The **GRC** in `GRCh38` is short for the **Genome Reference Consortium** and the task/mission of this consortium is to

> improve the human reference genome assembly, correcting errors and adding sequence to ensure it provides the best representation of the human genome to meet basic and clinical research needs. (source)

Faq

What does the `p13` in `GRCh38.p13` mean?

p is short for **patch** and the number after **p** tells the **patch number**. Thus `p13` means that we are looking at the `13th` patch release of the `GRCh38` reference assembly. Normally, patches add **minor changes** that **does not change the coordinate system**.

Note: These are also called **minor release**.

`GRCh38.p13` can be read as: **Genome Reference Consortium Human Build `38` patch release `13`** (source)

Note. the latest patch is `.p14` and it was issued in **2022**.

You can find more questions and answers at the Genome reference consortium FAQ page

Task

Every time there is a new release, often a **release note** is issued. It is a good idea to read these release notes to learn about the changes. For example, please go and read this release note about the `GRCh38.p13` release. You can also check the definitions for **major release**, **minor release**, different **patches** and their differences.

Note

The chromosome coordinate changes between two major release. We can see this when we use biological databases. For example, let's look at the *HBB* gene location in the NCBI GENE database:

- `chr11:5225464-5227071` in `GRCh38.p14`
- `chr11:5246694-5248301` in `GRCh37.p14`
- `chr11:5284832-5286439` in `T2T-CHM13v2.0` (source).

The minor releases often do not change the chromosome coordinates.
🔧 There are **tools** that help in **converting genome coordinates** between two different genome assemblies/builds major releases. One of which is called LiftOver. There are a few more bioinformatics tools for such conversions. You can learn about them here.

Task

Practice using LiftOver tool by converting the *PTEN* coordinate in `GRCh38.p13` that is `chr10:87863625-87971930` to `GRCh37.p13` and check whether the result is similar to what is reported in the NCBI GENE (i.e., `chr10:89623382-89731687`).

This concludes our brief introduction to the HGB and different human genome references. Now, let us shift our focus to the variation in the genome.

Variation in the genome

By **variation**, we mean **differences** in **DNA sequences** between any two individuals. A specific **site** or **locus** where this difference is observed is commonly referred to as a **variant**.

Some relevant concepts

- A locus with variation may have `2` or more different versions. Each version is called an **allele**.
  - A **biallelic** locus has `2` observed alleles across many samples in a cohort, one of which is the allele seen in the reference.
  - A **multiallelic** locus has `3` or more observed alleles across many samples in a cohort, one of which is the allele seen in the reference.
- The allele that is **similar** to the **reference genome** is called **reference allele**.
- The allele(s) that is/are **different** from the **reference genome** is/are called the **alternative allele(s)**.
- **Allele count** quantifies the occurrence of a specific allele within a population.
- **Allele number** quantifies the total number of different alleles present at a specific genetic locus.
- **Allele frequency** is the relative frequency of an allele at a particular locus in a population. In other words: Allele frequency = Allele count / allele number.
- The **most common allele** in a **population/cohort** is called the **major allele**. But this does not mean that it should be the **reference allele**!
  - Remember that the human reference genome was made from a handful of people and a large part of the human genome primary assembly came from one individual (please also remember that the later releases/builds have tried to address this issue).
- A **minor allele**, is an allele that is **less common** (compared to the major allele) in a **population/cohort**.
- When we talk about alleles, the concept of **genotype** also pops up! What does **genotype** mean? Humans are **diploid** organisms meaning they have **two copies**

of each autosomal chromosomes (each inherited from one of the parents). A genotype determines the `2` alleles at a particular locus each inherited from one of the parents.

> - A person has a **homozygous reference** genotype for a locus when both of her alleles at that locus are **similar** to the **reference allele**.
> - A person has a **homozygous alternate** genotype for a locus when both of her alleles at that locus are **similar to each other** but **different** from the **reference allele.**
> - A person has a **heterozygous** genotype for a locus when she has **two different alleles** at that locus, one of which could be the **reference allele**.

- Another concept related to the concept of genotype, is the concept of **phenotype** (aka **trait**). Phenotype is the physical properties and characteristics (e.g., pea color in Mendel's experiments) that are manifested by an organism. **Genotype** as well as the **environmental factors** contribute to the phenotype.

Example

Biallelic locus at position `2`.

```
1| 2 |345          # position
C| G |GTA          # reference
C|G/A|GTA          # sample 1
C|G/G|GTA          # sample 2
C|G/G|GTA          # sample 3
C|G/A|GTA          # sample 4
C|A/A|GTA          # sample 5
   ⇑
```

Multiallelic locus at position `4`.

```
123| 4 |5          # position
ATG| G |A          # reference
ATG|G/G|A          # sample 1
ATG|G/T|A          # sample 2
ATG|T/T|A          # sample 3
ATG|G/G|A          # sample 4
ATG|G/-|A          # sample 5
    ⇑
```

Now, that we know about a few concepts, let us learn about a source of variation in the genome and how it occurs.

Mutation as a source of variation

- One major source of variation is **mutation** that causes a permanent change to a DNA sequence.
  - Studies suggest that the mutation frequency in the human genome between generations typically ranges from 70 to 100 new single nucleotide variant mutations per generation. (Spielmann et al, 2018).
- Other sources of variation in a population are **recombination** and **gene flow** or **immigration of genes**.

How do mutations occur?

- One way for a mutation to occur is that during **DNA replication** an **error** occurs (i.e., a wrong nucleotide gets incorporated) and **proofreading** and **DNA repair machinery** fail to repair it. These mutations are known as *De novo* mutations. They are also known as *spontaneous mutation* as they occur in the absence of **mutagens** which are substances or agents that can induce genetic mutation.
  - `1` in 10^3 mutation occurs in the **absence** of **proofreading** and **mismatch repair**
  - `1` in 10^5-10^6 mutation occurs in the **absence** of **mismatch repair**
  - `1` in 10^8-10^9 mutation still remains after **proofreading** and **mismatch repair**
- Another way for a mutation to occur is via **DNA damage** for example caused by mutagens (e.g., Ultraviolet and X-ray radiation, or certain chemicals).
  - e.g., Ultraviolet radiation type A (UVA), often results in C → T mutations.

Mutations and the cell type

- These variations can occur in two types of cells:
  - germ cells or gametes (i.e., egg and sperm cells) resulting in **germline variants**.
  - somatic cells (i.e., all other cells that are not germ cells) resulting in **somatic variants**.
- **Germline variants** can be **passed on** to the **next generation** but this is not the case for the somatic variants.
- Somatic cells inherit the **same germline variants** as the original germ cells and **accumulate additional variants** throughout their lifespan.

Types, locations, and consequences of variations

Here, we explain a few different types of variation, where in the genome they may occur, and what are some possible consequences of such variations.

Types

Here, we enumerate a few types of variation.

Single nucleotide variant (SNV) (aka **substitution** or **point mutation**) occurs when a single nucleotide undergoes a change.

Example

```
AACGGTA          # original sequence
||| |||
AACCGTA          # with one nucleotide change
```

Note

Another terms that may be used interchangeably with SNV is **single nucleotide polymorphism (SNP)** provided that it is **common** in a population (i.e., allele frequency is ≥1%). But, for example, the Human Genome Variation Society (HGVS) recommends using SNV instead of SNP.

> In some disciplines the term **"mutation"** is used to indicate "*a change*" while in other disciplines it is used to indicate "*a disease-causing change*". Similarly, the term **"polymorphism"** is used both to indicate "*a non disease-causing change*" or "*a change found at a frequency of 1% or higher in a population*". To prevent this confusion we do not use the terms mutation and polymorphism (including SNP or Single Nucleotide Polymorphism) but neutral terms like **"variant"**, **"change"** and **"allelic variant"**.
> source

🎓 Thus, one way to choose how to refer to a change in the genome can be based on our discipline and the type of research/work we are involved in.

Insertion occurs when **one or more** nucleotides are inserted into a specific region of the genome.
Note that according to HGVS, single-base insertion is also an SNV.

Example

Insertion in `AACGGTA`

```
AACG-GTA        # original sequence
|||| |||
AACGAGTA        # with 1 nucleotide insertion
```

```
AACG---GTA      # original sequence
||||   |||
AACGACCGTA      # with 3 nucleotides insertion
```

Deletion occurs when **one or more** nucleotides are deleted from a specific region in the genome..
Note that according to HGVS, single-base deletion is also an SNV.

Example

Deletion in `AACGTGTA`.

```
AACGTGTA        # original sequence
|||| |||
AACG-GTA        # with 1 nucleotide deletion
```

```
AACGTGTA        # original sequence
|    |||
A----GTA        # with 4 nucleotides deletion
```

Insertions and deletions are also referred to as **indels**.

There are also **larger variants** (in terms of the number of nucleotides) called **structural variants** such as duplication, deletion, inversion...

Locations and molecular consequences

Mutations can occur either in the **coding regions** or **non-coding regions**. Here, we go through a few that occur in the coding region. Even though we do not cover variants occurring in the non-coding region of the genome, they can have an effect (e.g., if they occur in places that are important for the regulation of gene expression) and thus they are also important to know about and study.

Example

Consesequnces of SNVs on the exon one of the *HBB* gene coding sequence.

Normal

```
atg gtg cat ctg act cct [gag] ... ctg tgg ggc #codons
MET VAL HIS LEU THR PRO [GLU] ... LEU TRP GLY # amino acids
```

Silent mutation

```
                        ▾
atg gtg cat ctg act cct [gaa] ... ctg tgg ggc #codons
MET VAL HIS LEU THR PRO [GLU] ... LEU TRP GLY # amino acids
```

Missense mutation

```
                        ▾
atg gtg cat ctg act cct [gtg] ... ctg tgg ggc #codons
MET VAL HIS LEU THR PRO [VAL] ... LEU TRP GLY # amino acids
```

Nonesense mutation

```
                                    ▾
atg gtg cat ctg act cct gag ... ctg [tga] ggc #codons
MET VAL HIS LEU THR PRO GLU ... LEU [ ■ ] GLY # amino acids
```

- If a change in the codon sequence results in **no change** in **amino acid**, it is called a **synonymous** or **silent mutation**.
- If a change in the codon sequence results in a **change in amino acid**, it is called a **missense mutation**. Since the amino acid changes, it is a type of **non-synonymous mutation**.

- If a change in the codon sequence results in a **premature introduction of stop codon**, it is called a **nonsense mutation** (aka **stop-gain** mutation). This is also a **non-synonymous mutation**.
- If a change in the codon sequence results in change of a **stop codon to another amino acid**, it is called a **stop-loss mutation**.
- There are also other mutations such as **start-gain** and **start-loss**.
- When an **indel** has a length that is a **multiple of 3 nucleotides** (e.g., 3 or 6), it leads to an **in-frame indel**.
- When an indel's length is **not a multiple of 3 nucleotides** (e.g., 1, 2, or 4), it causes a **frameshift**, shifting all codons after the indel (see example below).

Example

```
        ▾▾▾
the ape ate the egg raw  # original codons
the ape     the egg raw  # deletion of "ate" (in-frame deletion)
```

```
 ▾▾ ▾
the ape ate the egg raw  # original codons
tpe     ate the egg raw  # deletion of "hea" (in-frame deletion)
```

```
    ▾▾
the ape ate the egg raw  # original codons
the eat eth eeg gra w    # deletion of "ap"  (frameshit deletion)
```

Variants can be either **neutral** with no effect, **beneficial**, or **detrimental**. For example, sickle-cell disease is caused by a single missense mutation that results in a single amino acid change. The same mutation confers resistance to malaria (Kwiatkowski, 2005). As for another example, an early nonsense mutation (i.e., stop-gain) can result in a truncated protein that cannot function properly.

In the **clinical context**, to prevent **ambiguity**, a set of words from a **controlled vocabulary** is used to refer to the **clinical significance** of variations. These are *benign*, *likely benign*, *uncertain significance*, *likely pathogenic*, and *pathogenic*. We will learn more about the benefits of using controlled vocabularies in different contexts in a future chapter.

Task

Use this codon table and

1. Find a single nucleotide change that results in a **silent mutation** in `CAC`.
2. Find a single nucleotide change that results in a **nonsense mutation** in `AAA`.
3. What is the result of a change from `TAA` to `TAG`?
4. In the sickle cell disease/anemia, GAG changes to GTG. Which amino acid is converted to which amino acid here?

Variant calling and bioinformatics tools

Let us assume that we have obtained a DNA sample from a person, performed high-throughput DNA sequencing (e.g., whole genome sequencing (WGS) or whole exome sequencing (WES) which is the sequencing of only protein-coding regions), and aligned the sequence reads to the human reference genome.

One thing that this data allow us to do is to **identify variants** i.e., to find the regions in this person's genome that are different from the human reference genome. This process is referred to as **variant calling**.

There are **bioinformatics tools** for calling variants which are called **variant callers**.

Earlier, we differentiated between **germline** and **somatic variants**. Accordingly, there are two types of variant callers (i.e., germline variant callers and somatic variant callers).

Each tool may use different approaches to identify a variant.

In the case of **germline variant calling**, a very basic approach would be to:

- Count the number of reads at a locus that support the **reference allele** (ref)
- Count the number of reads at that locus that support the **alternate allele** (alt)
- Calculate the **alternate allele fraction** as: $alt/(alt + ref)$
  - If the alternate allele fraction ~ `0` call a **homozygous reference** variant
  - If the alternate allele fraction ~ `0.5` call a **heterozygous** variant
  - If the alternate allele fraction ~ `1` call a **homozygous alternate** variant

Example

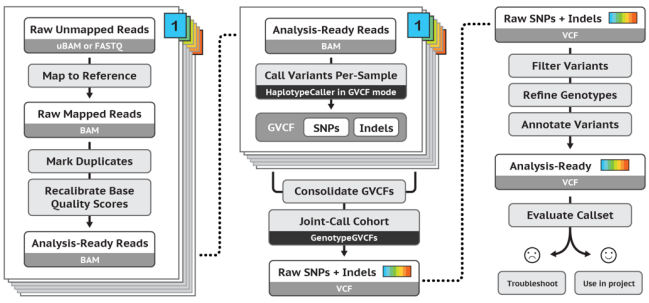
A simplified example of germline variant calling.

- `3` reads supporting the reference, and
- `3` reads supporting the alternate allele
- With an alternate allele fraction of `0.5`, we can conclude that this locus contains a **heterozygous** variant.

```
                                ⇓
00000000011111111112222222|2|22333333333344444444445555555556 #10's
12345678901234567890123456|7|89012345678901234567890123456890 #1's
CTAGAGCGTGGCCCGGAGCTGCCCTT|T|CCTCTTCGGTGAAGTTTTTAAAAGCTGCTGCGA #Ref
            CCGGAGCTGCCCTT|T|CCTCTTCGGTGA
                      CCTT|A|CCTCTTCGGTGAAGTTTTTAAA
               GAGCTGCCCTT|T|CCTCTTCGGTGAAGT
                   TGCCCTT|A|CCTCTTCGGTGAAGTTTTT
          CCCGGAGCTGCCCTT|T|CCTCTTCGGTG
                        T|A|CCTCTTCGGTGAAGTTTTTAAAAGC
                          ⇑
```

Note that this is a simplified example, and in real-world scenarios, variant calling requires a higher read depth and the consideration of additional factors such as base quality and mapping quality.

A commonly used variant caller is the Genome Analysis Toolkit (GATK), capable of identifying both germline and somatic variants. For example, the figure below illustrates GATK's workflow, starting from raw unmapped reads stored in FASTQ files and concluding with a list of **germline variants** stored in a VCF file.



As we see from the GATK example, variant callers store the result of variant calling in a standard file format known as Variant call format (VCF) file format.

After obtaining a list of variants, several analyses can be performed, including:

- Determining whether a variant is coding or non-coding
- Identifying known variants
- Identifying variants likely associated with rare diseases
- Assessing the type, molecular consequence, and clinical significance of each variant
- Filtering out variants that are common in a specific human population or are likely neutral, especially when dealing with numerous loci
- Prioritizing the remaining variants based on molecular consequence and clinical significance to focus on those most relevant to the study.

There are bioinformatics tools and well as biological databases that we can use to achieve these purposes. Here, we list only a few as examples.

- BCFtools can be used to manipulate VCF files or their binary counterpart which is BCF.
- vcfanno, ANNOVAR and a few other similar tools can be used for annotating the variants in a VCF file.
- Ensembl Variant Effect Predictor (VEP) determines the effect of variants on genes, transcripts, and protein sequence, as well as regulatory regions.
- Sorting Intolerant From Tolerant (SIFT) can predict whether an amino acid substitution affects protein function based on sequence homology and the physical properties of amino acids.
- Polymorphism Phenotyping v2 (PolyPhen-2) is a tool which predicts possible impact of an amino acid substitution on the structure and function of a human protein using straightforward physical and comparative considerations.

> **Note**
>
> Researchers also create **bioinformatics pipelines** that are composed of different tools which together perform all the tasks required for variant calling such as sequence read quality control, alignment, variant call, variant annotation, variant effect prediction, variant prioritization, etc.

Further reading

- You can read this Nature Scitable article about genetic mutation to consolidate some of the material that what you have learned in this chapter.
- To collect the material for this chapter, we got ideas from this online 2-hour course offered by the European Bioinformatics Institute (EBI). There are still parts in this 2-hour course that you can read to further your knowledge of the variation in the genome.

Chapter 6. Database similarity search and sequence alignment basics

In chapter 5, we learned about genome variation. We learned that variations in the germline can be passed on to the offspring. These changes at the molecular level are one source for evolutionary change. As the rate of change is not very large, if we look at two species with common ancestry, we expect that certain sequences to be similar, yet having some differences. For example, it is estimated than humans and chimpanzees diverged around `7`-`10` million years ago (White et. al., 2009). On the other hand, humans and rodents (including mice) diverged around `96` million years ago (Nei et. al., 2001). Thus, we would assume that if we look at sequence data (nucleic acid or protein) from these species, we should be able to see some similarities and some differences. Additionally, since there has been longer time since the divergence of the humans and the mice than the humans and the chimpanzees, we expect to see more differences between humans and mice than humans and chimpanzees. Let us take a look at the *sequence alignment* result of **hemoglobin subunit beta** (HBB) sequences across these `3` species to see whether our intuition is correct.
Knowing about what we just learned, one bioinformatics analysis task, is to **compare two or more sequences with each other** as we saw in the above example. We may ask what we would gain by such a comparison? Let us consider a few examples:

- For example, let us assume that we have obtained an unknown protein sequence. We could perform a *similarity search* and compare its sequence against a set of known protein sequences stored in a biological database. By doing this we can see whether it is already in a database (or in other words, we identify it), or what is known about it, or to which organism or organisms it belongs. We may be able to find out that our unknown protein belongs to a certain protein family or that it contains a certain conserved domain, or we could potentially assign a function to it (or a segment of it).
- As for another example, by comparing a set of sequences, bioinformaticians can find out how similar or different a set of sequences are and how they have diverged from each other over time. This is one of the subjects of study in the field of *comparative genomics*.

This was a rather long motivation to show a few of many possible reasons why we as bioinformaticians perform *sequence alignments* and *similarity searches*. But one question that we could ask is how these tasks are done? In what follows, we learn about the basic ideas and concepts.

Example

Comparing the sequence of **hemoglobin subunit beta** (HBB) across 3 species: Homo sapiens (humans), Pan troglodytes (chimpanzees), and Mus musculus (mice).

Looking at the *multiple sequence alignment* result using a tool called *Clustal Omega*, we can see that Homo sapiens and Pan troglodytes have the exact same HBB protein sequence while the Mus musculus's sequence is `19.73`% different from the other two.

```
1↝ sp|P68871|HBB_HUMAN
2↝ sp|P68873|HBB_PANTR
3↝ sp|P02088|HBB1_MOUSE

1↝ MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK 60
2↝ MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK 60
3↝ MVHLTDAEKAAVSCLWGKVNSDEVGGEALGRLLVVYPWTQRYFDSFGDLSSASAIMGNAK 60
   *****  **:**:.****** ********************:*:******: .*:*** *

1↝ VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG 120
2↝ VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG 120
3↝ VKAHGKKVITAFNDGLNHLDSLKGTFASLSELHCDKLHVDPENFRLLGNMIVIVLGHHLG 120
   ********: **.*** ***.******:**********************::* **.**:*

1↝ KEFTPPVQAAYQKVVAGVANALAHKYH 147
2↝ KEFTPPVQAAYQKVVAGVANALAHKYH 147
3↝ KDFTPAAQAAFQKVVAGVATALAHKYH 147
   *:*** .***:********.*******
```

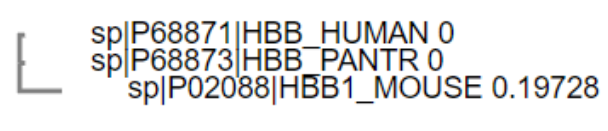
- A `*` shows amino acids that are conserved across these 3 species.
- A `:` shows amino acids with *strong* similar properties.
  - e.g., Threonine (T) and Serine (S) are both *polar* amino acids.
- A `.` shows amino acids with *weak* similar properties.
  - e.g., Alanine (A) is *hydrophobic* while Cysteine (C) is *special* which has sulfur in its side chain.
- An empty space shows amino acids that are not similar.

Percent identity matrix

```
                          HBB_HUMAN  HBB_PANTR   HBB1_MOUSE
   1↝ sp|P68871|HBB_HUMAN   100.00     100.00       80.27
```

```
2⇝ sp|P68873|HBB_PANTR   100.00     100.00       80.27
3⇝ sp|P02088|HBB1_MOUSE   80.27      80.27      100.00
```

Using this matrix (or a similar one), it is possible to draw a *phylogenetic tree* and **cluster** protein sequences from different species based on their similarities and differences. We can see that the more similar species in terms of their of HBB protein sequence cluster together.



Note that, what we did here is called **multiple sequence alignment** because more than `2` sequences were aligned against each other. If we align only `2` sequences against each other, it is known as **pairwise alignment**.

Sequence identity and similarity

When comparing two sequences (nucleic acid or protein) against each other, we often encounter two concepts: *sequence identity* and *sequence similarity*. They measure how **identical** or **similar** two sequences are respectively. When comparing nucleic acids, *sequence identity* and *sequence similarity* can be used interchangeably. But in the context of protein sequence comparison, these two have their own meaning. Below, we will learn about them via a few examples.

Example

Nucleic acid sequences

```
Sequence identity: 100% (10/10)
AACGGTACCT      # sequence 1
AACGGTACCT      # sequence 2
**********

Sequence identity: 90% (9/10)
AACGGTACCT      # sequence 1
AACGGTACTT      # sequence 2
******** *

Sequence identity: ?
        Gap-excluded identity (gaps are excluded): 100% (8/8)
        Gap-compressed identity (consecutive gaps are counted once):
~89% (8/9)
AACGGTACCT      # sequence 1
AAC--TACCT      # sequence 2
***  *****
```

Protein sequences

```
Sequence identity: 100% (10/10)
Sequence similarity: 100% (10/10)
AVTALWGKVN  # sequence 1
AVTALWGKVN  # sequence 2
**********

Sequence identity: 90% (9/10)
Sequence similarity: 100% (10/10)
        - because T & S are strong similar amino acids
AVTALWGKVN  # sequence 1
AVSALWGKVN  # sequence 2
**:*******

Sequence identity: 90% (9/10)
Sequence similarity: 90% (9/10)
AVTALWGKVN  # sequence 1
AVMALWGKVN  # sequence 2
** *******
```

Now, that we know how sequence identity and similarity are calculated, let us focus on the question of how *sequence alignment* is done.

Sequence alignment

There are two forms of sequence alignment: *local alignment* and *global alignment*.

If we are interested in finding **the most similar regions/segments** between two sequences, we may use a **local alignment algorithm**. This is because a local alignment algorithm, such as the **Smith-Waterman algorithm**, **starts by locating the region/segment that is the most similar** between the two sequences being aligned and **expands the alignment around this region**. If both sequences share a similar region (or regions that might be biologically important), local alignment is likely to find it.

On the other hand, instead of starting from the most similar region, a **global alignment algorithm, takes into account both sequences in their entirety**, compares them, and finds the alignment such that **the number of aligned nucleotides or amino acids is maximal**. Thus, instead of starting from the most similar region, a global alignment algorithm, such as the **Needleman-Wunsch algorithm**, **starts from the beginning** of the two sequences being aligned and **expands the alignment (sometimes by introducing gaps) until it reaches to the end** of one of the sequences.

Let us see an example, where `2` sequences undergo local and global alignment and see how the results differ.

Note

Even though local and global alignments are used for different purposes, if the sequences are significantly similar, the alignment result may be identical.

Note

Remember that we first heard about **dynamic programming** in **chapter 4**, when we were explaining the alignment of short sequencing reads to a reference genome. Both the **Smith-Waterman** and the **Needleman-Wunsch** algorithms are dynamic programming algorithms for solving the sequence alignment problem.

Briefly, when using a dynamic programming algorithm, we **break down a complex problem** into **simpler sub-problems**. If we can find solutions for the sub-problems, we should be able to find the solution for the main problem. Note that not all problems can be solved by this technique unless they satisfy certain

properties such as having optimal substructure and overlapping sub-problems (Here, we do not get into these details).

Example

Local versus **global** alignment when aligning the following `2` sequences (`40` and `50` nucleotides long).

```
CCTGTTTCCAGGGCCCTCTTGCTTACTGTATAGTGGTGTC # 40 nucleotides long

GTGACTTTTGGATTTTTGCCAGGGCCCTCTTGCTTACTGTATAGTGGTGC # 50 nucleotides
long
```

Here, we can see that how these two algorithms produce different results as explained above.

```
             3 TGTTTCCAGGGCCCTCTTGCTTACTGTATAGTGGTG     38
               |.||.|||||||||||||||||||||||||||||||     # Local
            14 TTTTGCCAGGGCCCTCTTGCTTACTGTATAGTGGTG     49

1 -----------CCTGTTTCCAGGGCCCTCTTGCTTACTGTATAGTGGTGTC   40
             ..|.||.|||||||||||||||||||||||||||||.     # Global
1 GTGACTTTTGGATTTTTGCCAGGGCCCTCTTGCTTACTGTATAGTGGTGC-   50

# PARAMETERS
## Needleman-Wunsch                ## Smith-Waterman
## Gap open 10                     ## Gap open 10
## Gap extend 0.5                  ## Gap extend 0.5
```

Note: We learn about these parameters below. Also **note** that even though the main part of the difference is due to how these algorithms work, part of the difference can be due to the differences in scores/penalties used.

Both the Smith-Waterman and the Needleman-Wunsch algorithms employ similar concepts and components, including *substitution matrix*, *gap penalty*, *scoring matrix*, and a *traceback/backtrack process* to perform local and global alignment respectively. We explain each of these below by **focusing on the local alignment and the Smith-Waterman algorithm** since it is the type of alignment used by tools such as BLAST (short for *basic local alignment search tool*).

Substitution matrix

A substitution matrix is a matrix that keeps the scores assigned to each possible substitution between `2` nucleotides or `2` amino acids. These scores, for example in the case amino acid substitution, can be calculated by using the information about how amino acids are substituted with each other over time when species evolve.

Example

An example of nucleotide **substitution matrix** where simply a **match** (i.e., a nucleotide is substituted by itself) gets a score of `2` and a **mismatch** (i.e., a nucleotide is substituted by another nucleotide) gets a score of `-1`. In this example, all match (or mismatch) scores are similar as a result of assuming that the substitution frequencies are equal.

```
   A  G  C  T
A  2 -1 -1 -1
G -1  2 -1 -1
C -1 -1  2 -1
T -1 -1 -1  2
```

An example of amino acid **substitution matrix** known as **BLOSUM62** (source). BLOSUM is short for *blocks substitution matrix*. `62` in BLOSUM62 tells that protein sequences used to create this matrix had approximately `62%` sequence identity.

```
A  4
R -1  5
N -2  0  6
D -2 -2  1  6
C  0 -3 -3 -3  9
Q -1  1  0  0 -3  5
E -1  0  0  2 -4  2  5
G  0 -2  0 -1 -3 -2 -2  6
H -2  0  1 -1 -3  0  0 -2  8
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4
K -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5
M -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
```

A negative substitution score (e.g., `-4` for Proline (P) → Phenylalanine (F)) tells that such a substitution may have a considerable effect on a protein's structure and function such that it is not encouraged by natural selection.

Note that in addition to the `BLOSUM62`, there are many other substitution matrices such as **PAM** (short for *point accepted mutation*) and **PET** (short for *pair exchange table*). Different substitution matrices are tailored to **detect similarities among sequences** that are **diverged by differing degrees**. For example, `BLOSUM62` substitution matrix is used when aligning proteins that we expect to have less than `62%` sequence identity.

More-info

If you are interested in learning more about `BLOSUM62` substitution matrix, please read *Where did the BLOSUM62 alignment score matrix come from?*

Gap penalty

When aligning two sequences, sometimes we need to **introduce gaps**. Each gap represents the **occurrence of an insertion or a deletion** of a nucleotide or amino acid into or from one of the two sequences during evolution. In order to keep the sequence alignment result useful and meaningful, the number of gaps and the length of gap segments should be kept to a reasonable degree. That is why a gap penalty is used. Every time a **new gap** is introduced to the alignment, a *gap-opening penalty* is used. Similarly, every time **an existing gap gets extended**, a *gap-extension penalty* is used. By setting different penalty values, we can affect the number of gaps and their length. For example, if we set the gap-opening penalty to be higher than the gap-extension penalty, we are favoring the extension of existing gaps than introducing new gaps.

Scoring matrix

Scoring matrix is used to keep track of the scores we obtain from solving each of the sub-problems. Focusing on the Smith-Waterman algorithm here, when aligning `2` sequences `A` of size `n` (i.e., $A = a_1a_2 \dots a_n$) and `B` of size `m` (i.e., $B = b_1b_2 \dots b_m$), the scoring matrix is going to have `n+1` rows and `m+1` columns (e.g., if $A = ACTG$ and $B = ACTG$, our scoring matrix is going to be 5×5). The first row and the first column of the scoring matrix are filled in with `0`s . We fill the remaining elements of this matrix by calculating the H_{ij} for each element using the following algorithm:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \geq 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \geq 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \qquad (1 \leq i \leq n, 1 \leq j \leq m)$$

If we set the *gap-opening* and the *gap-extension* penalties to be equal, the above algorithm will become simplified:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ H_{i-1,j} - W_1, \\ H_{i,j-1} - W_1, \\ 0 \end{cases}$$

where:
$s(a_i, b_j)$ is the score obtained by consulting the substitution matrix.
$H_{i-1,j-1} + s(a_i, b_j)$ is the score of aligning a_i and b_j
W_k is the penalty of a gap with length k
$H_{i-k,j} - W_k$ is the score if a_i is at the end of a gap of length k
$H_{i,j-l} - W_l$ is the score if b_j is at the end of a gap of length l
W_1 is the cost of a single gap.
0 means there is no similarity up to a_i and b_j.

As we can see, each element of the scoring matrix is filled in by calculating `3` scores and choosing the maximum non-negative value. If all these `3` scores are negative, H_{ij} is set to `0`.

> **Note**
>
> We will be using the simplified version in this chapter.

Traceback/backtrack process

The purpose of the traceback process is to determine the alignment that gives us the highest score. The **first step** in the traceback process in the Smith-Waterman algorithm involves **finding the element in the scoring matrix with the highest score**. Next, we look at the **neighbors** of this element (i.e., neighbor to the left, above, and on the diagonal) and we select the **one with the highest score** and **mark the transition between these two elements with a proper arrowhead** (i.e., `↖`, `←`, or `↑`). If two or all three neighbors have similar score, we can choose any of them. We **continue** with this procedure until we **reach an element with score zero**. This is where the traceback process ends. Next, we could use the **path** generated by the traceback process and the direction of the arrowheads to write down the highest scoring local alignment.

- If an arrowhead is pointing diagonally (i.e., `↖`), we have either a **match** or a **mismatch**.

- If an arrowhead is pointing to the left (i.e., $\leftarrow$), we introduce a **gap** in the sequence A.
- If an arrowhead is pointing upward (i.e., $\uparrow$), we introduce a **gap** in the sequence B.

More-info

If you want to practice more with the Smith-Waterman algorithm, you may use this tool from the University of Freiburg.

Example

Local alignment using the Smith-Waterman algorithm

Let us use the nucleotide substitution matrix we saw in the example above i.e., we use `2` for a *match*, `-1` for a *mismatch*. Also, let us use `1` for *gap*.

Aligning `ACTG` and `ACTG`

```
                    S     A  C  T  G     S     A  C  T  G
ACTG   sequence A      0  0  0  0  0        0  0  0  0  0
ACTG   sequence B   A  0  2  1  0  0     A  0  ↖  1  0  0
****                C  0  1  4  3  2     C  0  1  ↖  3  2
                    T  0  0  3  6  5     T  0  0  3  ↖  5
                    G  0  0  2  5  8     G  0  0  2  5  ↖
```

Aligning `ACTG` and `GTCA`. Remember that we start from highest score and stop when we encounter `0`. Here, there are `4` possible starting points and right after one backtrack step, we encounter `0`. So there are `4` possible local alignments.

```
                    S     G  T  C  A     S     G  T  C  A
G    T     C    A      0  0  0  0  0        0  0  0  0  0
G or T or  C or A   A  0  0  0  0  2     A  0  0  0  0  ↖
*    *     *    *    C  0  0  0  2  1     C  0  0  0  ↖  1
                    T  0  0  2  1  1     T  0  0  ↖  1  1
                    G  0  2  1  1  0     G  0  ↖  1  1  0
```

Aligning `ACTG` and `ACG`. Note that here a gap (shown by `-`) was introduced to achieve the best alignment.

```
                    S     A  C  G     S     A  C  G
ACTG   sequence A      0  0  0  0        0  0  0  0
AC-G   sequence B   A  0  2  1  0     A  0  ↖  1  0
** *                 C  0  1  4  3     C  0  1  ↖  3
                    T  0  0  3  3     T  0  0  ↑  3
                    G  0  0  2  5     G  0  0  2  ↖
```

Aligning `ACG` and `ACTG`.

```
                    S     A  C  T  G     S     A  C  T  G
AC-G    sequence A     0  0  0  0  0        0  0  0  0  0
ACTG    sequence B  A  0  2  1  0  0     A  0  ↖  1  0  0
** *                 C  0  1  4  3  2     C  0  1  ↖  ←  2
                    G  0  0  3  3  5     G  0  0  3  3  ↖
```

Database similarity search and BLAST

Now, that we have learned about the basics of local sequence alignment using Smith-Waterman algorithm, let us consider the task of **database similarity search**. Let us assume that we have a nucleotide or amino acid sequence and we want to find similar sequences that are stored in a database. An intuitive approach would be to perform sequence alignment between our sequence (aka. the *query*) and every other sequence that is in a database (aka. the *subjects* or *targets*), **calculate** an alignment score, **normalize** the scores so that they can be compared, and **select/display** those that have the highest score (i.e., having high similarity). Even though this makes sense intuitively, we have to take into account the amount of time this takes.

The Smith-Waterman algorithmic complexity is $\sim O(nm)$ where n and m are the lengths of the sequences A and B that are being aligned respectively. If we search a large database for similarity, using this algorithm requires a large amount of time. Luckily, tools using **heuristic methods** have been developed to address this challenge. A heuristic method solves a problem by finding a solution that is **good enough** or **satisfactory** for a particular purpose but **not necessarily perfect or optimal**.

Basic Local Alignment Search Tool **(BLAST)** is an example of a heuristic method that addresses the time challenge when searching a large database for similar sequences. According to an estimate BLAST is ~`100` times faster than if we would solely rely on the Smith-Waterman algorithm. However, we should be aware that it is not guaranteed that BLAST produces as accurate alignments as those that can be produced by the Smith-Waterman algorithm. Nonetheless, BLAST is a widely used tool for searching for sequence similarity.

The main idea behind the BLAST's approach is that if two sequences are similar, a short segment of the *query sequence* should **match** a segment of the *subject sequence* either perfectly in the case of nucleotide sequence or rather well in the case of protein sequence. Thus, the query sequence is divided into short subsequences called a **word**. These words are searched for in the subject sequence. A match provides a **seed** for BLAST to **extend** by performing a **sequence alignment** with the aim of maximizing the segment alignment score, resulting in a **high-scoring segment pair** (HSP). In the end, BLAST lists all HSPs sorted based on their scores.

Note

Note that based on the purpose of our search and the chosen BLAST program, different **word sizes** are set and this size has an effect on the **sensitivity** as well as the **speed** of the BLAST.

✓ **Check**

In this chapter, we do not cover different BLAST tools such as `blastn`, `blastp`, `blastx` and their use cases. But you can start learning about them by reading *Guide to BLAST home and search pages*.

Task

Before moving forward, read this guide describing the BLAST search result report.

Here, we learn more about the outputted result and especially about the *Bit score* and the *Expect value* (aka. E-value).

BLAST Glossary, defines the ***bit score*** and ***Expect value*** as follows:

> The bit score, S', is derived from the **raw alignment score**, S, taking the statistical properties of the scoring system into account. Because bit scores are **normalized** with respect to the scoring system, they **can be used to compare alignment scores from different search results**.

We need to **normalize** the scores before comparison because factors such as the **length of the sequences** may **affect the alignment scores**. For example, a longer sequence may get a better alignment score than a shorter one only because it is longer and not necessarily better. The **bit score** indicates the **goodness of an alignment**. The **higher** the bit score, the **better** the alignment.

> The **Expectation value** or **Expect value** represents the **number of different alignments** with **scores equivalent to or better than** S that is expected to occur in a database search **by chance**. The lower the E-value, the more significant the score and the alignment.

If we perform a protein sequence search (see the following example) and check the *Alignments* tab in the report page, we see the results of similarity search with some details. Let us look at one of these results.

- The upper sequence marked by *Query* is the sequence that we used for our search.
- The lower sequence marked by *Sbjct* is a sequence from the database that aligned to our query.
- The sequence in between, shows the alignment result.
  - A match is marked by a letter that is similar to both the query and the subject.
  - A mismatch is marked by an empty space (i.e., ` `).
  - A strong similar mismatch/substitution is marked by a plus sign (i.e., `+`).
  - A gap is marked by a hyphen (i.e., `-`).
- The ***Identities*** refers to the *sequence identity*. Since there are only `3` amino acids between the query and the subject that are different, the **sequence identity** is $47/50 = 0.94$.
- The ***Positives*** is a term used by BLAST to refer to *sequence similarity*. We saw that there are `3` different amino acids in the result. In the following result, these are marked by a `+` that means that they are **strong similar substitutions**. Thus, the **sequence similarity** is $50/50 = 1$.
- We see that there has been no need to introduce a gap as the ***Gaps*** column is `0`.
- Score section is shown as `102 bits(254)` where the **bit score** is `102` and the **raw score** is in **parenthesis** (i.e., `254`). A bit score of `102` means that, on average, we would have to score 2^{102} (or $\approx 5 \times 10^{30}$) pairs of sequences, before we see a raw score $S \geq 254$ **by chance**.
- The ***Expect*** (i.e., E-value) for this example is very small (i.e., $3e$-26) which means that the likelihood of getting an alignment score of `102 bits(254)` or better just by random chance, is very low. Thus, we could be confident that the alignment result is not spurious.

Example

An example of BLAST alignment result.

```
------------------------------------------------------------------
Score                  Expect ... Identities   Positives
Gaps
102 bits(254)   3e-26  ... 47/50(94%)   50/50(100%)     0/50(0%)
------------------------------------------------------------------

Query  1   VHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLST  50
           VHLTPEEK+AVTALWGKVNVDEVGGEALGRLLVVYPWTQRFF+SFGDLS+
Sbjct  2   VHLTPEEKNAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFDSFGDLSS  51
```

Note

- To make effective use of BLAST, it is advisable to have a clear understanding of the meaning behind each available option and the fundamental concepts employed by the BLAST algorithm. Thus, it is recommended to read and consult the BLAST topics and the NCBI's BLAST Glossary pages.
- `BLOSUM62` is used by default by the *basic local alignment search tool* (BLAST) algorithms for scoring amino acid alignment. NCBI's BLAST substitution matrices page suggests which substitution matrix to use for different purposes.
- When using the **Web BLAST**, under the *Algorithm parameters*, we can set an *E-Value* threshold. This determines which alignments will be reported. A higher *E-Value* threshold is less strict.
- Originally, BLAST has set the *E-value* threshold to `10` with the aim of ensuring that no biologically significant alignment is missed.
- If an *E-value* is less than $1e\text{-}180$, NCBI BLAST displays it as `0`.
- An alignment with a bit score > 50 could be considered a good alignment.

Multiple sequence Alignment (MSA)

At the very beginning of this chapter, when we were inspecting the HBB protein across `3` different species, we learned that MSA involves alignment of more than `2` sequences. We saw the results from Clustal Omega (aka. `ClustalΩ`, or `Clustal O`) which is a tool for MSA. We can use the MSA, for example, if we have a set of sequences and seek to identify **conserved sequence segments** among them. These segments may point out that the sequences are related in terms of evolution. In the example, we saw that the entire HBB protein were conserved between the Homo sapiens and the Pan troglodytes and we saw that Mus musculus had an `80.27`% sequence identity with the other `2` sequences. Even though we aligned only `3` sequences, with the Clustal Omega, this tool can be used to align hundreds or more sequences.

Essentially, an MSA algorithm conducts pairwise sequence alignment among all sequences within a set, using resulting alignment scores to infer similarity between pairs and to order the sequences in the MSA result.

More-info

If you are interested to learn more about the Clustal Omega's algorithm, please read the *Clustal Omega* section of the Wikipedia entry on Clustal tool.

Info

FASTA file format

When using tools such as *BLAST* or *Clustal Omega*, a file format that is often used to provide the sequences is the **FASTA file format**. This is a text-based file format for storing and representing either nucleotide or amino acid sequences.

An entry in a FASTA file has `2` parts:

- A **header/description** section that starts with `>`
- The **sequence** section that can be spread on more than `1` line.

For example, the following shows the content of a FASTA file with the `4` sequences, `3` of which we used for the MSA example at the beginning of the chapter:

```
>sp|P68871|HBB_HUMAN Hemoglobin subunit beta OS=Homo sapiens OX=9606
GN=HBB PE=1 SV=2
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK
VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG
KEFTPPVQAAYQKVVAGVANALAHKYH
>sp|P68873|HBB_PANTR Hemoglobin subunit beta OS=Pan troglodytes
OX=9598 GN=HBB PE=1 SV=2
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK
VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG
KEFTPPVQAAYQKVVAGVANALAHKYH
>sp|P02088|HBB1_MOUSE Hemoglobin subunit beta-1 OS=Mus musculus
OX=10090 GN=Hbb-b1 PE=1 SV=2
MVHLTDAEKAAVSCLWGKVNSDEVGGEALGRLLVVYPWTQRYFDSFGDLSSASAIMGNAK
VKAHGKKVITAFNDGLNHLDSLKGTFASLSELHCDKLHVDPENFRLLGNMIVIVLGHHLG
KDFTPAAQAAFQKVVAGVATALAHKYH
>sp|P02091|HBB1_RAT Hemoglobin subunit beta-1 OS=Rattus norvegicus
OX=10116 GN=Hbb PE=1 SV=3
MVHLTDAEKAAVNGLWGKVNPDDVGGEALGRLLVVYPWTQRYFDSFGDLSSASAIMGNPK
VKAHGKKVINAFNDGLKHLDNLKGTFAHLSELHCDKLHVDPENFRLLGNMIVIVLGHHLG
KEFTPCAQAAFQKVVAGVASALAHKYH
```

To learn more about the FASTA file format, please read its Wikipedia entry here.

Further reading

- To collect the material for part of this chapter, we got ideas from the *Bioinformatics for beginners* book written by Supratim Choudhuri (more info about the book here) as well as these videos made by the NCBI BLAST.
- You can access BLAST documentation here. It has many useful links and HowTos as well as videos.
- The BLAST Sequence Analysis Tool chapter from the *The NCBI Handbook*, and BLAST QuickStart: Example-Driven Web-Based BLAST Tutorial chapter from *Comparative Genomics* books provide some further information and examples about BLAST.

⁂

Chapter 7. Transcriptome and gene expression profiling

Over the past few chapters, we have learned about the human genome, DNA sequencing methods and technologies, and the Human Genome Project that sequenced the human genome. We also learned about variations in the genome and some of their consequences and how we could use high-throughput sequencing data, together with a few bioinformatics tools (e.g., quality control tools, sequence aligners, and variant callers) to identify such changes in the genome.

We know that practically every cell in our body carry the same genome (ignoring the few nucleotide changes that they have acquired along the way e.g., as a result of DNA replication or environmental factors). Yet, we see that we have different cells with different functions. For example, skin cells, muscle cells, and nerve cells all look different and function differently. So, we may ask what results in such a big difference?

When a **gene is transcribed** in a given cell, it can be said that that **gene is expressed** in that cell. We may have noted that some genes such as the *PTEN* gene is **expressed differently** across different tissues (it is expressed the most in the human brain tissue and the least in the human liver tissue, e.g., see Figure 1). Could this observation leads us to an answer? Could an answer be that different cell types use and express different genes or different amounts of the same genes and that is why they are different? These are very interesting observation and questions and that is why, this chapter, we **shift our attention** from studying the genome to the **studying of the transcriptome**.

Figure

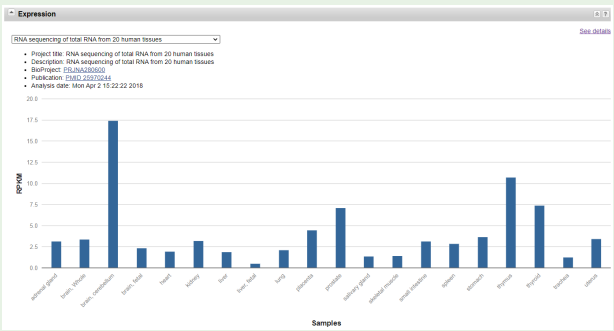


Figure 1. *PTEN* expression in different tissues. Figure is a screenshot from the *PTEN* entry from the NCBI Gene database.

We define the *transcriptome* as *the collection of all coding and non-coding transcribed DNA sequences in an organism*. There, by *transcribed DNA sequences*, we meant the **RNA molecules** that result from **copying the DNA sequences** with the help of **RNA polymerase enzymes**. And by *coding*, we meant those subset of RNA molecules that are further **translated into proteins** whereas *non-coding* refers to RNA-molecules that are **not translated into protein**, yet they may have a function in the cell (e.g., ribosomal RNA or rRNA is a non-coding RNA which acts as the primary building block for ribosomes). Figure 2, illustrates the steps involved in the expression of a protein-coding gene.

Figure

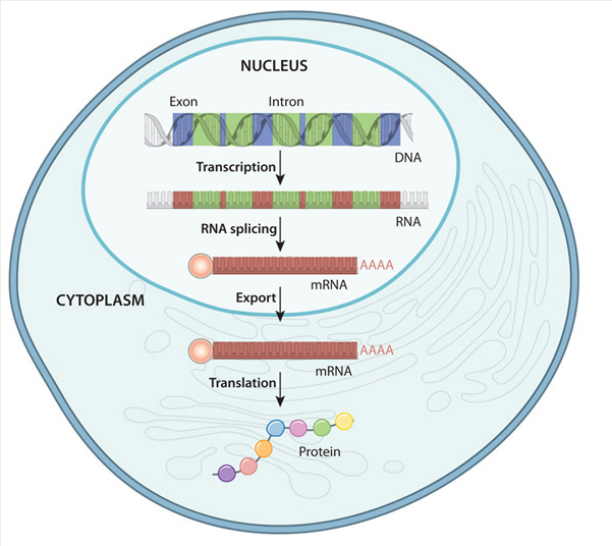


Figure 2. Different steps in the expression of a protein-coding gene (Figure source).

Here, we learn about a few approaches that bioinformaticians use to **study and analyze the transcriptome and gene expression**. We also learn about a few other **applications** of the studying the transcriptome.

To **study the transcriptome** of a cell (or in other words to perform **transcriptomics studies**), a way would be to check **what genes are expressed and by how much** at a **given point in time** and **under a particular condition**. In other words, we could perform a **gene expression analysis** (aka. gene expression profiling) experiments.

In the **past**, the expression of only **a few genes** could be measured **at a time** by techniques such as the northern blot technique. However, thanks to the advent of high-throughput sequencing methods, we now can practically detect and measure practically every expressed gene in a bulk of cells or even in a single cell (single-cell approaches are discussed in a future chapter).

High-throughput gene expression profiling

Different methods for high-throughput gene expression profiling exist such as DNA microarrays, serial analysis of gene expression (SAGE), cap analysis gene expression (CAGE) and RNA-seq. In this chapter, we briefly learn about two of them, namely the **DNA microarrays** and the **RNA-seq**.

DNA microarray

DNA microarray method was developed in 1980s-1990s. DNA microarrays can be used for many **different applications** such as **gene expression profiling**, **SNP genotyping**, and **DNA methylation profiling**. However, here, we focus on the use of DNA microarrays in gene expression profiling. With the microarray technology, we can **measure** the **relative expression** of **hundreds to thousands of genes simultaneously**.

A DNA microarray (aka. **DNA chip**) is a **slide** with a set of **microscopic spots**. Each spot contains a set of identical, short, single-stranded DNA molecules called **probes**. Each spot is used to measure the expression of one particular gene. Note that this means **we know already which gene we are going to measure** by a particular spot and this information is **encoded** in the **nucleotide sequences in that spot** which are **complementary** to a portion of the **gene of interest**. Also note that there are **different DNA microarray providers/vendors** such as Affymetrix, Agilent, and Illumina and each have their **own way of producing their microarrays** (and especially the spots on the microarrays).

Here, we describe a **two-channel microarray gene expression experiment**. In such an experiment, RNA molecules are **collected/extracted/isolated** from two types of samples: a **reference** (aka. **control**) and a **case** (aka. **test**). The **control** sample could be cells under a **normal condition** (e.g., cells not treated by a drug or healthy cells from a tissue) whereas the **case** sample could come from similar cell type but it has been under a **different condition** (e.g., treated with a drug or cells obtained from a tumor tissue). The RNA molecules are, then, converted into **complementary DNA (cDNA)** by **reverse transcription** (note that, here, the use of the primer determines which type of RNA is going to be converted into cDNA. If **polyT primers** are used, **only mRNA** molecules are converted to cDNA whereas if **random primers** are used, **all types of RNA** molecules are converted). Control and case samples are also **tagged with fluorescent labels/dyes with two different colors** (e.g., the cDNA molecules in the **control** sample are labeled with a **green fluorescent dye** (e.g., cy3 dye) whereas the cDNA molecules in the **case** sample are labeled **red** (e.g., cy5 dye)). Next, case and control samples are **pooled/mixed together**, **denatured** to obtain single-stranded cDNAs, and **added to the microarray**. The **probes** on the microarray then **capture cDNA molecules** that are **complementary** to their

sequences via **hybridization** (i.e., the cDNA molecules and probes that are complementary to each other, bind to each other). After the hybridization step, the microarray is **washed** to remove **unhybridized molecules** and dried. Next, a **laser** is shone with **two different emission frequencies** on the microarray to excite the **fluorescent dyes** and to **make them glow**. The **light intensity** of a spot is **proportional** to the **number of cDNA molecules that hybridized to that spot**. **For each emission frequency**, an **image is taken** by scanning the microarray using a **microscope**. These two images are, then, **merged** by putting them on top of each other. After the merge, if a spot appears to be **more red than green**, this tells that the expression of the gene corresponding to that spot is **higher in the case than in the control**. If a spot appears to be **more green than red**, tells that the expression of the gene corresponding to that spot is **higher in the control than in the case**. If a spot appears to be **yellow**, this tells that **both case and control have equal expression** (see Figure 3). The **light intensity** can be **quantified**, **normalized**, and used as a **measure of the expression of a gene**. Finally, we could **analyze the data** to find similarities and differences between the two samples in terms of gene expression profile.

Figure

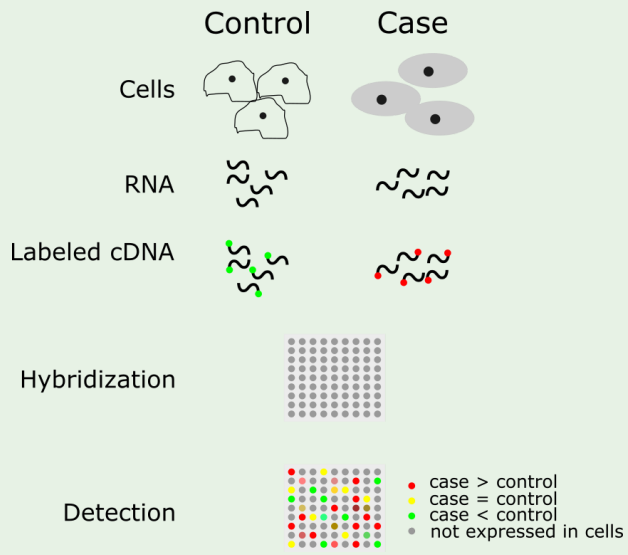


Figure 3. Schematic illustration of gene expression profiling via two-channel DNA microarray. Note that the items in this illustration are **not to scale**!

RNA-seq

RNA-seq is another high-throughput gene expression profiling method that **relies on sequencing**. Using this method, we can measure the expression of any part of the genome that is transcribed (in addition to its application in gene expression profiling,

RNA-seq can be used for other purposes, some of which we learn about later in this chapter).

In this method, we start by **extracting** the **RNA molecules**. Similarly to the microarray experiment, depending on our application, we can decide whether we want to focus only on mRNAs or whether we want to profile all classes of RNA molecules (aka. total RNA). Next, **RNA molecules** are **fragmented** and **converted into cDNA** by **reverse transcription** (**note** that we could also, first, convert the RNA molecules into cDNA and then fragment them but this may introduce a bias. For example, see Figure 3a in (Wang, 2009)). **Sequence adapters** are, then, **ligated** to the cDNA molecules. Depending on the amount of the starting material, we may decide to perform PCR amplification. Next, **size selection** step selects cDNA fragment with length within the preferred ranges of the sequencing technology. Finally, **sequencing** is performed. **Paired-end sequencing** is **preferred** since e.g., it facilitates the **determination of the transcript structure** (See Figure 4, ❷ and ❸).

> **Note**
>
> The **library preparation protocol** could be either ***non-stranded*** (sometimes called *standard protocol*) or ***stranded*** (aka *strand-specific* or *directional*). In the non-stranded protocol, we **loose the information** about the **orientation** of the **transcripts** while the stranded protocol preserves this information. However, the non-stranded protocol is cheaper and suitable for many applications such as gene expression analysis. For more information, check e.g., this research article.

Once we have the **sequencing reads**, the next step is to perform **quality control**, take proper actions depending on the result of the quality control (i.e., preprocess the data), and **align the sequencing reads** to a **reference**. We briefly learn about the RNA-seq sequence data alignment in the following section.

RNA-seq sequencing read alignment

Let us start by checking what **types of sequencing reads** we may obtain if we extract/capture **only mRNAs** and perform an RNA-seq experiment. To do this, let us consult the Figure 2 again. From that figure, we see that in the **mature mRNA**, there are **no intronic sequences** and different exons have been spliced together. Also, the **mature mRNA** has a **polyA tail on its 3´ end**. As a result, we will have the following **3 types of sequencing reads**:

- Reads that fall within one particular exon (aka. **exonic reads**. See Figure 4, ❶).
- Reads that span two exons (aka. **junction reads**. See Figure 4, ❸).
- Reads that come (partially) from the polyA tail (aka. **polyA end reads**).

- We do not see the polyA portion of such reads e.g., in the IGV since they are **often removed/trimmed in the quality control and preprocessing step and before the alignment step**. Also, note that the polyA sequence is not part of the genome and that it was added only after the transcription (i.e., the addition of polyA tail is a **post-transcriptional modification**).
- Note that even though some bioinformaticians remove/trim polyAs in a sequencing read before alignment, these can be used to infer the **direction of transcription** as well as they can help to **resolve the 3´ end of overlapping transcribed regions** (Wang, 2009).

Figure

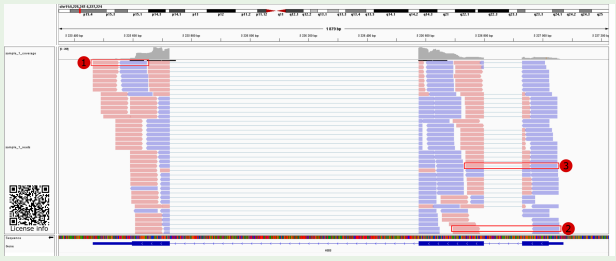


Figure 4. IGV screenshot of paired-end RNA-seq sequencing data at the *HBB* locus. Note that we do not see reads mapping to the introns.
❶ shows a pair of reads where both are aligned to the same exon.
❷ shows a pair of reads where each of the reads are aligned to different exons.
❸ shows a pair of reads where one of the reads in the pair is aligned to one exon and the other read is aligned to 2 different exons.

Note that paired-end reads similar to ❷ and ❸, are useful in determining the structure of the transcript.

Now that we know about the possible types of reads we can obtain from an RNA-seq experiment, let us consider the **task of sequence alignment**. Remember from chapter 5 that the purpose of sequence alignment is to find out the genomic locations of the sequencing reads against a reference. Here, we have the **option** to **choose** either a **reference genome** or a **reference transcriptome** (provided that they are available, if no reference is available, we could **assemble the reads** and construct transcripts. This is known as *de novo assembly*, but we do not go into its details here).

We already know about the reference genome from **chapter 5**. The **transcriptome reference**, on the other hand, is constructed by **collecting** the **sequences** of **all the known transcript variants**.

In case we use a **reference genome**, the ***junction reads*** **cannot be aligned to the reference genome directly** and we need to **introduce gaps** in our alignment or in other words, we have to allow **gapped alignment** (aka. **spliced alignment**. In **chapter 6**, when learning about the Smith-Waterman algorithm, we saw that if there was insertions or deletions in one of the two sequences being aligned, we would introduce gaps. A similar idea is used here). On the other hand, if we use a **reference transcriptome**, aligning the junction reads is **simpler** as there is **no need for gaped alignment**.

Even though aligning the reads against a transcriptome reference may be simpler, but we have to consider the possibility that our sample expresses **transcript variants** that are **not in the transcriptome reference**. In such a case, we **fail to detect and quantify such novel transcripts**. As a result, **if finding novel transcript variants is important in our study, we would better align our data to the reference genome**. Thus, typically, RNA-seq reads undergo alignment against the reference genome instead of transcriptome reference.

There are different bioinformatics tools available such as HISAT (or its newer version HISAT2) and STAR for aligning of the RNA-seq sequencing reads.

> ⭐ **Cool-fact**
>
> HISAT2 makes use of a method that is developed by a group of Finnish scientists, namely, Jouni Sirén, Niko Välimäki, and Professor Veli Mäkinen.

Once the sequencing reads from an RNA-seq experiment is aligned, we can proceed to the downstream analysis. Next, we learn about the *RNA-seq gene expression quantification and normalization*.

RNA-seq gene expression quantification and normalization

There are **different approaches** in **quantifying** the **expression** of the genes from RNA-seq data. A **simple approach** for **quantifying a gene** would be to **count the reads that fall in (or overlap with) each of the exons that are annotated to belong to that gene.** Once we have the **counts** for each exon, we could **sum them up** to get a **total count for that gene**. Note that this approach does not take into account different transcript variants/isoforms of the genes being quantified. Instead of exons, we could use the transcripts. Here, exons or transcripts are referred to as **features**.

Now that we know about one possible approach, we also need to answer some relevant questions. The first question we need to answer is: **from where** should we **obtain information** (such as **coordinates information**) about **the features**? One

recommended source is the GENCODE that provides us with **gene annotation information** for human and mouse genome.

The next question is about the **counting policy**. This is because the way we count the aligned reads affects the final results we obtain (e.g., should we count a **junction read twice** or only once? Should we count a read that only **partially overlap** with a feature? What if a sequencing read overlaps `2` different features? For an example, please refer to the figure under this HTSeq tool tutorial page). However, if we are consistent with our decisions when quantifying different samples, we could assume that such decisions affect the samples in a similar fashion and thus the samples are comparable with this regard.

The **output** of the **quantification** step is a **data matrix** where, typically, **each row is a gene or a transcript** and **each columns is a sample** (in real life examples, a data matrix may have `20000` rows or more; See Table 1. Also **note** that sometimes **other columns** may be added to the data matrix e.g., a column with the transcript length).

If we look at Figure 5, we see the **coverage track** of two samples at the *HBB* gene locus. We see that `sample_1` has much more sequencing reads aligned to the *HBB* gene than the `sample_2`. After the quantification step, we would assume that the data matrix would look like what is shown it Table 1 where `sample_1` has a larger count than `sample_2`. But, can we, right away, say that the *HBB* gene is expressed more in `sample_1` than `sample_2`? What if, when we were preparing the sequencing libraries, we had more starting material in `sample_1` than in `sample_2`? What if, `sample_2` had a lower quality and part of the RNA molecules have been already degraded? Would these have an effect on what we see here? To address these and other issues/questions (e.g., **RNA-seq biases**), and to be able to **reliably compare** genes within a sample or samples against each other, we **need to normalize the data in the data matrix**. Different ways to normalize the count data from an RNA-seq experiment have been developed by bioinformaticians. Each of these normalization approaches have **their own merits** but some have become **more adopted** than the others especially **depending on the application**. To name a few, we can mention:

- Reads Per Kilobase of transcript per Million mapped reads (**RPKM**).
  - $$RPKM = 10^9 \times \frac{Reads\ mapped\ to\ transcript}{Total\ reads \times Transcript\ length}$$
  - This is a **within-sample normalization method** that will remove the **feature-length** and **library-size effects**.
  - See the `Y-axis` of the bar plot in **Figure 1** and you can see that this normalization approach has been used.
- Fragments Per Kilobase of transcript per Million mapped reads (**FPKM**)
  - This is a derivative of RPKM.
- Transcripts Per Million (**TPM**)

- $$TPM = 10^6 \times \frac{Reads\ mapped\ to\ transcript / Transcript\ length}{Sum(Reads\ mapped\ to\ transcript / Transcript\ length)}$$
  - This is a **between-sample normalization method** that normalizes for the **differences in composition** of the transcripts.
- Median of Ratios (MOR)
  - This is a **between-sample normalization method**.

Note

⚠ **Note** that for **certain downstream analyses** such as *differential gene expression analysis*, the use of **RPKM** or **TPM** is **not recommended**.

Figure

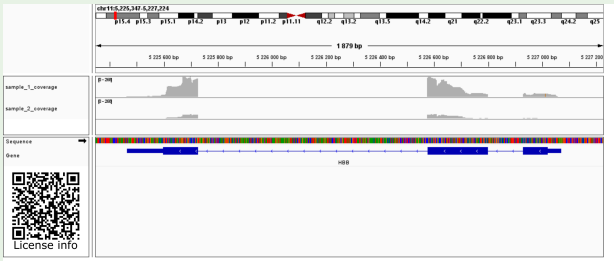


Figure 5. IGV screenshot of 2 samples' coverage tracks from paired-end RNA-seq sequencing data at the *HBB* locus.

Table

Table 1. A toy example of an $n \times m$ data matrix that results from the quantification step. Here, for example, in `sample_1`, `500` reads/fragments have been aligned to the *HBB* gene.

	$sample_1$	$sample_2$	...	$sample_j$	...	$sample_m$
HBB	500	100	...	...	...	...
...	...	...	...	...	...	...
$gene_k$	...	...	...	...	...	...
...	...	...	...	...	...	...
$gene_n$	...	...	...	...	...	...

Let us see how we can **normalize** our data matrix using the **TPM normalization approach** via the following example.

Example

An example of the TPM normalization with a toy 2×2 data matrix (i.e., `2` genes and `2` samples).

Data matrix before normalization.

gene	gene length (kb)	sample_1	sample_2
gene_1	2	4	8
gene_2	5	10	22

Step ❶. Divide the counts in the data matrix with gene length in kilobases to get **RPK** (reads per kilobase).

gene	gene length (kb)	sample_1	sample_2
gene_1	2	$4/2 = 2$	$8/2 = 4$
gene_2	5	$10/5 = 2$	$22/5 = 4.4$

Step ❷. Add together all the RPK values to get a **sample-specific scaling factor**.

gene	sample_1	sample_2
Scaling factor	$2 + 2 = 4$	$4 + 4.4 = 8.4$

Step ❸. Divide each RPK value with the **scaling factor in millions**.

gene	sample_1	sample_2
gene_1	$2/(4/10^6)$	$4/(8.4/10^6)$
gene_2	$2/(4/10^6)$	$4.4/(8.4/10^6)$

TPM normalized data matrix

gene	sample_1	sample_2
gene_1	500000.00	476190.48
gene_2	500000.00	523809.52

Note: Since our toy example has only `2` genes, it may seem that dividing the sample-specific scaling factor by 10^6 does not make sense! But in a real data matrix, summing up the RPK values of around `20000` genes (as we did in step ❷) results in a large number and dividing it by 10^6 makes sure that scaling factor is not large and subsequently resulting in very small normalized values.

Note: In this toy example, even though before normalization it seemed that genes in `sample_2` were expressed twice as much as the genes in `sample_1`, after TPM normalization, samples seem to have more or less similar expression.

Note. Our data matrix before normalization contained **discrete numbers** resulted from counting the reads. After the TPM normalization, the values in the data matrix were transformed to **continuous numbers**. Some downstream analysis methods may require to work with discrete values and in such a case, TPM normalization is not usable/suitable.

Task

Can you tell why in step ❶ of the TPM normalization, we divided the counts by the **gene/transcript length** in kilobases?

After proper normalization of the gene expression quantification, our data is **ready for the downstream analyses**. In the following sections, we briefly learn about a few of these analyses, namely *differential expression analysis* and *cluster analysis*.

Differential expression analysis

Differential expression (DE) analysis, as its name suggests, attempts to **detect genes** that are **expressed at different levels across two groups of cells** (e.g., cells treated with a drug vs. cells treated with a placebo or cells with under different conditions e.g., healthy vs. diseased).

Before the advent of high-throughput techniques, DE analysis was done one or a few genes at a time using e.g., Northern blotting (e.g., see *Figure 5* in Korc et al., 1992). Thanks to the high-throughput techniques such as RNA-seq, DE analysis can be performed on all the genes that are detected in an experiment. Differential expression analysis is **the most common type of RNA-seq data analysis**.

Let us take a look at Figure 6. There, we see that $gene_k$ in the samples under `condition_2` are expressed around twice as much as the samples under `condition_1`. Does this mean that $gene_k$ is differentially expressed across the `2` conditions?

Figure

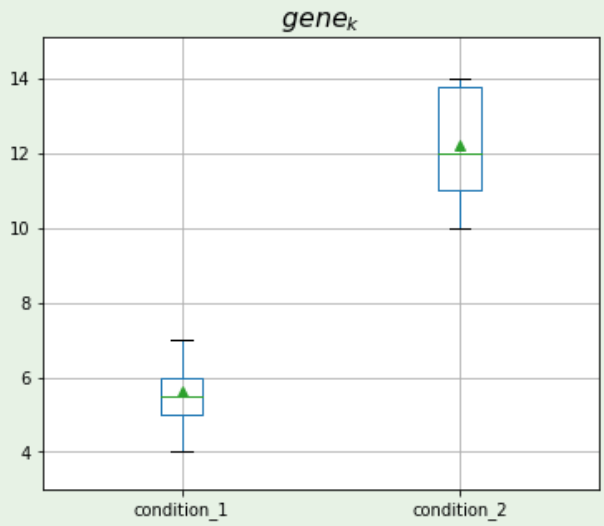


Figure 6. Toy example shows the expression of $gene_k$ in `20` samples (`10` samples in each condition/group). Samples under `condition_2` are expressed around twice as much as samples under `condition_2`. Is this gene differentially expressed across the `2` conditions?

In the boxplot, the green triangles show the condition/group means and the green lines show the condition/group medians.

```
## Data underlying this figure is as follows:
condition_1 = [6, 6, 6, 5, 4, 5, 7, 5, 7, 5]
condition_2 = [10, 11, 14, 11, 12, 13, 12, 14, 14, 11]
```

There are **different methods** developed to reliably answer the question above. A very famous bioinformatics tool for differential expression analysis is DESeq2 (one can tell the popularity of this method/tool by the number of the citations it has accumulated since its publication in 2014. The article had been cited `28267` times in November 2022 and it has gathered ~8000 more citations since November 2021). **Here, we**

focus on a simple approach and we do not go into the details, for example, how DESeq2 works.

To answer the above question, we could use a **statistical test** that takes into account the **difference between the group means (or group medians)** and tells **whether the difference is statistically significant**. For example, Wilcoxon rank-sum test takes into account the observed values from `2` groups (e.g., the gene expression values under `2` different conditions), calculates a **test statistic** and a **p-value**. Here, the **p-value** tells **how likely** it is to see a **test statistics as large or larger as was calculated from the observed values assuming both groups have the same median**.

If we perform a Wilcoxon rank-sum test on the data used to create the figure above (see example below), we see that it seems unlikely (p-value=`0.000157`) to see a **test statistic of** `-3.7796` **or larger** assuming both conditions have the **same median**. Often, **before we perform a test**, we decide on a **significance threshold** and if the **p-value** we obtain is **below this threshold**, we call the **result statistically significant**. The threshold historically has been set at `0.05` but in bioinformatics purposes it is **not uncommon** to see a threshold **as small as** `0.001` **or even smaller**. The procedure we described here is known as **hypothesis testing**. So, if we have set the significance threshold at `0.001`, the **statistical test result** (i.e., p-value=`0.000157`), suggests that **the two groups/conditions**, in our example, **does not have the same median** or in other words the **group medians are different**, and the **difference is statistically significant**.

Example

Performing Wilcoxon rank-sum test over data used to create Figure 6 using **Python**.

```
from scipy.stats import ranksums
condition_1 = [6, 6, 6, 5, 4, 5, 7, 5, 7, 5]
condition_2 = [10, 11, 14, 11, 12, 13, 12, 14, 14, 11]
ranksums(condition_1, condition_2)
# Output: RanksumsResult(statistic=-3.7796, pvalue=0.000157)
```

⚠ There is one **issue** that **we have to be aware of**! If we set the significance threshold at `0.05` and we **perform** `100` **hypothesis tests (or statistical significance tests)** on `100` **genes that are not differentially expressed**, `5` **out of the** `100` tests may get a **p-value below the significance threshold just by chance**, and as a result, we mark them as **statistically significant** and **differentially expressed** even though in reality that is not a case. These are called **false positives** (💡 A lighthearted method to grasp the notion of a false positive test result is to consider a scenario where a **male individual** takes a **pregnancy test**, yielding a **positive outcome**—essentially

indicating that the male is pregnant. Of course, this outcome is false, given that biologically, males cannot experience pregnancy!). Now, consider a data set that has `20000` genes and we perform this many statistical tests with a significance threshold of `0.05`. A good portion of our **significant results** are going to be **false positives**. This is an issue that is known as the **multiple hypothesis testing problem** or **multiple comparison problem**. There are methods that try to address this issue and they are known as **multiple hypothesis testing correction methods**.

Task

How many false positive results would we expect to get if we perform `20000` statistical tests with a significance threshold of `0.05` and we know that all `20000` genes are not differentially expressed?

Once we have a reliable set of differentially expressed genes, there are multiple ways to proceed with our analysis. To mention a few, for example, we could check if these **differentially expressed genes** are **affecting a particular biological pathways** or we could **annotate them with Gene Ontology (GO) terms** and check whether they are **involved in a particular biological processes or whether they are localized in a specific component of a cell** and check whether this **reveals something interesting in terms of biology** (we will learn about Gene Ontology and biological pathways in **chapter 9**).

Cluster analysis

In addition to the differential expression analysis, we could perform **cluster analysis**. With this analysis, **genes with similar expression profiles** are **assigned to their own clusters** (or groups) and we check the clusters to see whether we could **recognize patterns in the gene expression profiles**. This approach makes sense, since, several **genes** that **belong to a particular biological pathway**, are **often co-regulated**. For example, we could **possibly infer a function** for the **gene product** of **an unknown gene** if it happens to **cluster** with a **group of genes whose functions are well-known** as we may suspect that our unknown gene belongs to a common pathway or having a similar function. For an example of clustering result look at *figure 1 A* in Ylipää et al. (2015).

Other applications of RNA-seq

Earlier, we mentioned that the RNA-seq can be used for purposes other than quantification of the gene expression. Here, we list a few of them.

Detection of variants

Since an RNA-seq experiment results in sequencing data, we can use it to **detect variations** in the **exonic regions** (provided that we have enough depth of coverage to be able to reliably call such variations). See Figure 7 for an example.

Figure

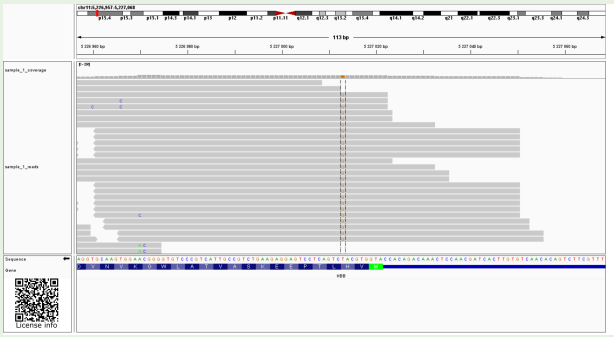
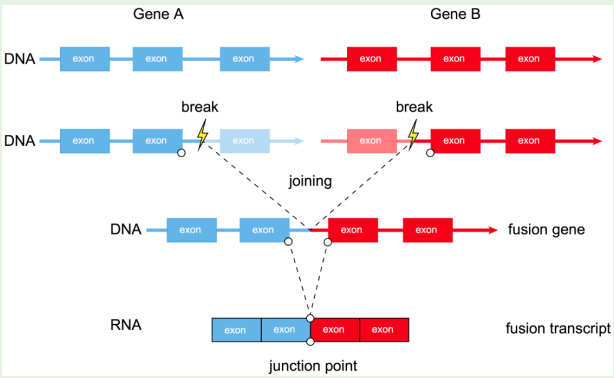


Figure 7. A possible variant at the *HBB* locus can be seen from paired-end RNA-seq sequencing data .

Detection of fusion genes

We could also use the RNA-seq data to look for fusion genes. **Gene fusion** can occur via **different mechanisms** one of which is the **deletion of a genomic region**. In gene fusion via a genomic deletion, the deletion causes that some of the exons from one gene to sit next to the exons in another gene on the same chromosome resulting in a viable fused gene (see Figure 8). We could use paired-end reads to detect fusion genes.

Figure



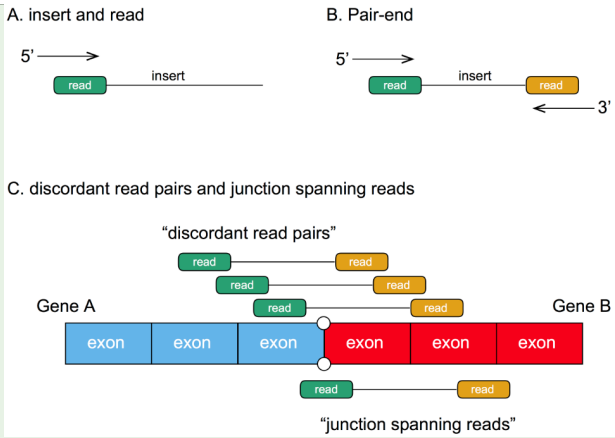


Figure 8. Schematic illustration of how a gene fusion occurs and how they can be detected from paired-end reads (Figure source).

Note. In the bottom figure, *insert* refers to a cDNA fragment that was sequenced from its both ends. A *discordant read-pair* is a **pair of reads** from the **same fragment/insert** whose alignments to the reference genome have **distance and/or orientation** that **differ from what is expected** if the entire fragment was contiguous on the reference genome.

For an example of fusion gene, let us consider the prostate cancer disease. In this disease, around half of the patients carry a fusion gene that is the result of the fusion of the *ERG* and the *TMPRSS2* genes (Tomlins et al, 2008, Jumar-Sinha et al, 2008; See Figure 9). This fusion results in the over-expression of the *ERG* gene.

Figure

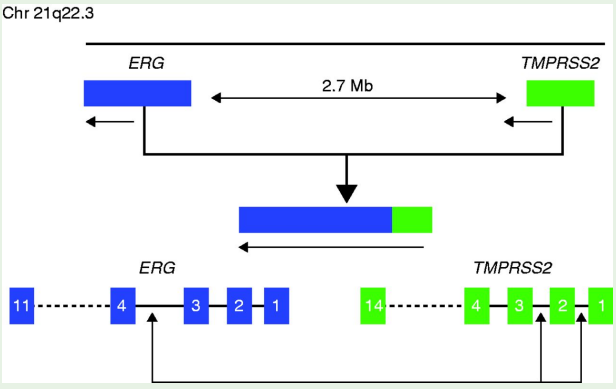


Figure 9. Schematic illustration of the TMPRSS2–ERG fusion (Figure source).

Discovery of novel transcript variants

When we were discussing the alignment of RNA-seq data, we also learned that we could use this data to **discover novel RNA transcripts** in our sample(s). These could be common or rare in a sample in terms of abundance.

> **Note**
>
> According to Mortazavi (2008), in case we want to detect and quantify all biologically-relevant RNA molecule classes in a sample, we need to sequence more than 40 million reads from that sample. To detect and quantify rare transcripts, we need to increase the number of sequenced reads.

Gene expression databases

Here, we name `2` databases that **store gene expression data**.

Gene Expression Omnibus (GEO) of the National Center for Biotechnology Information (NCBI)

> GEO is an international public repository that archives and freely distributes microarray, next-generation sequencing, and other forms of high-throughput functional genomics data submitted by the research community. source

Expression Atlas of the European Bioinformatics Institute (EBI)

> Expression Atlas is an open science resource that gives users a powerful way to find information about gene and protein expression. Our mission is to provide the scientific community with freely available information on the abundance and localization of RNA (and proteins) across species and biological conditions such as different tissues, cell types, developmental stages and diseases among others. source

> **Task**
>
> Based on what we have learned in this chapter, summarize the similarities and differences of the DNA microarray and RNA-seq as `2` high-throughput techniques for gene expression profiling.

Further reading

- *A survey of best practices for RNA-seq data analysis (2016)* is a good review article to learn about different steps and best practices involved in an RNA-seq experiment starting from experiment design up until the interpretation of different analyses' results.
- Misuse of RPKM or TPM normalization when comparing across samples and sequencing protocols reviews two of the normalization methods we learned about and discusses where it is suitable (or it is not suitable) to use them.

⁂

Chapter 8. Epigenome and gene expression regulation

In earlier chapters, we noted that the *PTEN* gene is **expressed differently** in **different tissues**, and we started to wonder whether different cells (e.g., nerve cells and muscle cells) are different because genes are expressed differently in them **even though they share practically the same genome**. If we assume that this is the case, the next question we could ask is that **what are the mechanisms** that result in such a behavior? Or in other words, **what regulates the expression of different genes in different cells?** Thus, in this chapter, we turn our attention to **gene expression regulation** and learn only a few of the available approaches that bioinformaticians use to study it.

The **gene expression regulation mechanisms** that we learn in this chapter are mainly **epigenetic changes**. Epigenetic changes are **modifications to the DNA** or **DNA-associated proteins** that ❶ **does not change the nucleotide sequence of the underlying DNA**, ❷ that they are **reversible**, and ❸ they can be passed on from a parent cell to its daughter cell in cell division and some of these changes may even be inherited by the next generation. **Epigenome**, thus, refers to the **collection of all such modifications across the genome**. And, **epigenomics** is a field of research that focuses on the **genome-wide characterization of such modifications**. Let us start by learning about the **packaging of the DNA**.

DNA packaging and chromatin accessibility

Each **human diploid cell** contains around `2` meters of DNA (that is approximately `2` nanometers wide). How does all this fit inside a **human cell's nucleus** which is around `10` to `20` micrometers in diameter (which is around one tenth of the thickness of a human hair)? The answer lies in **DNA packaging**. DNA packaging starts by the **tight wrapping** of the **DNA** around a set of proteins called **histones** forming a **DNA-protein complex** known as the **chromatin**. A **unit of chromatin** is known as **nucleosome**. A nucleosome is composed of approximately **146 base pairs of DNA** wrapped around a **protein octamer** that is composed of `2` of each `4` different **histone proteins** (i.e., **H2A**, **H2B**, **H3**, and **H4** also known as **core histones**). This packaging reduces the length by around `7` times. Next, the **chromatin** undergoes a few more times of **folding** and **looping** which results in the **chromatid** of a chromosome. DNA is in its **most condensed form** at the **metaphase** stage during mitosis which is approximately `10000` times shorter than unpacked DNA.

Info

The **tight wrapping** of **DNA** around **core histones** is achieved thanks to the **electrostatic interaction** between the **DNA** which is **negatively charged** (due to the **phosphate groups** in the DNA backbone) and **histone proteins** that are **positively charged**.

Due to the way DNA is packaged, it is possible that **some loci that encode for a set of genes** are **not tightly wrapped** around the core histones and thus they are **accessible** for the **transcription machinery** to bind to and start the transcription. These **relaxed loci**, that are **often under active transcription**, are known as euchromatin. On the other hand, some other loci are **tightly packed** and thus **inaccessible** for the transcription machinery to bind and to transcribe (see Figure 1). These **tightly packed (or condensed) loci** are known as heterochromatin. This way, **certain genes** have the potential to be **expressed** and certain genes are not going to be expressed (or in other words they are **turned off**).

Figure

Figure 1. Packaging of DNA by wrapping the DNA around histone proteins.

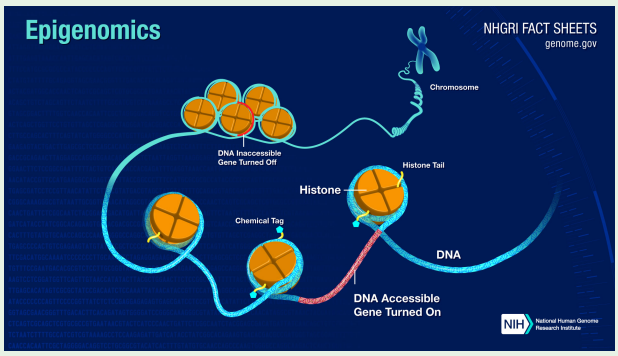


Figure source

Approaches to study genome-wide chromatin accessibility

Over the years, several assays/methods such as DNase-seq, FAIRE-seq, and ATAC-seq have been developed to study the **genomic accessibility** in a **genome-wide manner** using **high-throughput sequencing technologies**.

Currently, the ATAC-seq (assay for transposase-accessible chromatin using sequencing) method is more popular than the other methods as it is **faster** (it takes less than a day), **more sensitive**, and **requires less starting material** ($500 - 5 \times 10^4$

cells vs $10^5 - 10^6$ cells in other assays). Here, we briefly learn about the ATAC-seq method.

In the ATAC-seq method, hyperactive Tn5 transposase enzymes are used to **cleave accessible double-stranded DNA (dsDNA)** and to **tag** them with **sequencing adapters**. Next, **sequencing library is constructed** via **PCR-amplification** which is followed by **paired-end sequencing**.

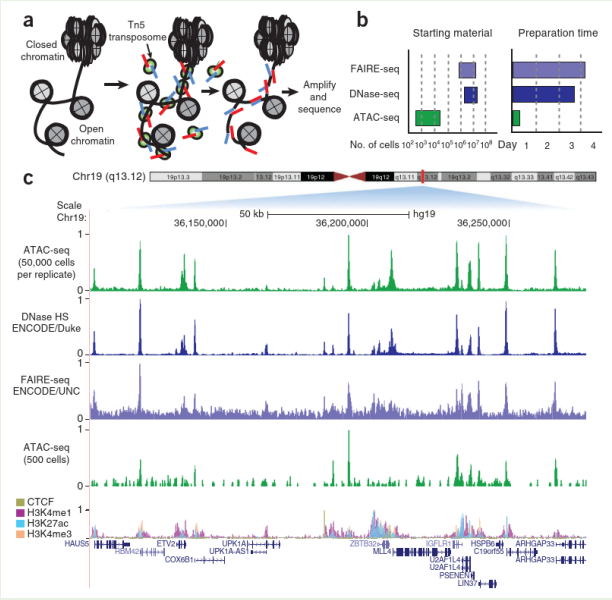
The resulting sequencing reads undergo **quality control** (e.g., using bioinformatics tools such as FastQC and more specifically **ataqv**) and then are **aligned** to a **reference genome**.

Bioinformatics tools such as Model-based analysis of ChIP-Seq (MACS) are then used to detect genomic regions having **enrichment of aligned sequence reads** when compared against the **background** (i.e., the inaccessible loci). Regions enriched with aligned sequencing reads are called **peaks** and they **mark** loci where the **chromatin** is **accessible** (see Figure 2).

Once we have the location of the peaks, we could use the available biological knowledge to annotate them using available bioinformatics tools. For example, we could use the `annotatePeaks.pl` program from the HOMER software to **associate detected peaks** with **genomic features** such as the distance to transcription start site (TSS) of the nearest gene, whether the peak overlap an intronic, exonic, untranslated region and other annotations.

Figure

Figure 2. ATAC-seq method and its comparison against other methods.



Note: Data shown here is from the application of the listed methods on

lymphoblastoid cells (which are transformed B-lymphocytes).
Figure source

We can, also, **quantify the extent of the accessibility** at each locus by simply counting the **amount of aligned sequencing reads** to that locus. However, we need to first **address and correct** a few **potential issues** introduced by a few **biases**. For instance, a locus may gather more sequencing reads (e.g., if a genomic duplication or amplification has occurred at that locus) or less sequencing reads (e.g., AT-rich and GC-rich regions may be underrepresented due to PCR-amplification bias). After the correction step, we can assume that we have obtained a reliable quantification. If we want to **compare** results from **different samples**, we also need to **properly normalize** the data. After proper normalization, it is possible to detect regions that are **differentially accessible across samples**, for example, samples from **different cell types** or sample under **different conditions** e.g., healthy vs. disease.

ATAC-seq data can be analyzed to reveal information other than the **genomic chromatin accessibility**. For example, ATAC-seq data can be used to infer the **position** and **packing** of the **nucleosomes**.

Histone modification

We learned that the **tight wrapping** of DNA around the core histones is achieved by the **electrostatic interaction** between the **DNA** and the **histone proteins**. What happens if the **histones** become **less positively charged**? We would expect that the chromatin to become less condensed at loci where **histones** are **less positively charged**. This is achieved by the addition of an **acetyl group** (i.e., CH_3CO) to a particular amino acid on the **histone tails** (aka. **N-terminal tails**). This process is known as acetylation. Following the same logic, the opposite reaction (i.e., **deacetylation**) that is the removal of the acetyl group from a particular amino acid at the histone tail results in the **condensation of the chromatin**. **Acetylation** has been associated with an **increase in transcription activity** whereas **deacetylation** has been linked with a **decrease in transcription activity**.

Acetylation and deacetylation are **not the only modifications** that can occur to different amino acids on the histone tails. There are many more modifications such as **methylation** (and **demethylation**) that occur to different amino acids on the histone tails. These are collectively known as the histone code (see Figure 3). To name a few, the **acetylation** of **lysine 27** on **H3 histone** (aka. **H3K27ac**) is a marker for **higher activation of transcription** whereas **trimethylation** of the **same amino acid** (i.e., **H3K27me3**) is a marker for **repression or down-regulation of the nearby gene**. **Tri-methylation** of **lysine 36** on **H3 histone** (aka. **H3K36me3**) is associated with the **transcription of active euchromatin** and typically found in the **body** of **actively**

transcribed genes, rather than at promoters. In Figure 2, we see 2 more modifications, namely H3K4me1 and H3K4me3.

- **H3K4me1**, similarly to H3K27ac, is a marker for **higher activation of transcription** and additionally it marks the gene **enhancers**. An **enhancer** is a **short DNA locus** in a close proximity of a particular gene (upstream or downstream) to which **activator proteins** may bind and by doing so it may **enhance** the **transcription** of that particular nearby gene.
- **H3K4me3** is a marker for **active chromatin** (i.e., euchromatin) and for **activation of transcription of the nearby genes**.

Figure

Figure 3. Schematic illustration of histone modifications collectively known as the histone code.

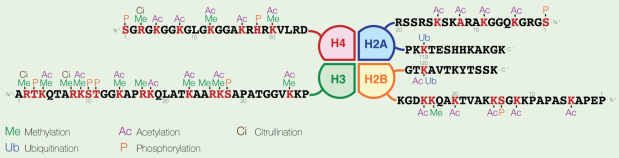


Figure source

We may ask what **CTCF** is in Figure 2? **CTCF** is a transcription factor (TF) that in humans is encoded by the *CTCF* gene. **Transcription factors** are **proteins** that **bind to a specific DNA sequence** and **control the rate of transcription**. CTCF, in addition to the **regulation of transcription**, is involved in other cellular processes such as insulation.

Now, that we know about the **regulation of transcription** via **histone modification** and **TF binding** (i.e., **protein-DNA interactions**), we may ask how bioinformaticians **measure** and **study** these events (to get data similar to what is shown by the lowermost track in Figure 2)? We find the answer to this question in the next section.

Chromatin immunoprecipitation (ChIP) assay

We can use **ChIP assays** to detect particular **protein–DNA interactions** (i.e., TF-DNA interactions) as well as particular **histone modifications** in a **genome-wide manner**. Two available high-throughput ChIP assays are ChIP-chip (or ChIP-on-chip) that uses **DNA microarrays** and ChIP-seq that uses **high-throughput sequencing**. Here, we briefly learn about ChIP-seq experiment.

The first step in ChIP-seq is to **cross-link** (see definition below) the DNA-protein complexes with e.g., formaldehyde. Next, the **sample chromatin** is **fragmented**. This step is followed by **chromatin immunoprecipitation**. **Immunoprecipitation** (IP) is a technique of **precipitating** (see definition below) a particular **protein** out of solution using an **antibody** that **specifically binds to that particular protein**. Thus, in this step, we use a **proper antibody** that is **specific** to the **protein (or TF)** of interest or **histone modification** of interest to **precipitate** it and as a result of this step **enrich** them in our sample. **Note** that since the protein and DNA are cross-linked, when we enrich the protein or histone modification of interest, we are also **enriching the DNA fragments that are bound (i.e., cross-linked) to the proteins of interest**. After the precipitation step, enriched DNA loci are **purified** by first **removing the cross-link** between the DNA-protein complex and **degrading the proteins** and **extracting the DNA**. Once we have the DNA fragments, we can proceed with the **sequencing library preparation** step (see Figure 4).

Definition

to cross-link. to **form** a **bond** or a **short sequence of bonds** that **links one polymer chain to another**.
definition source

to precipitate [*chemistry*]. to undergo or cause to undergo a process in which a **dissolved substance separates** from **solution** as a fine **suspension of solid particles**.
definition source

After the sequencing step, the obtained sequencing reads undergo **quality control**, **preprocessing**, and next, **aligned** to a **reference genome**. If looked at in a **genome browser** such as the IGV, we could spot **peaks** as a result of lots reads having aligned to the peak regions when compared to other regions (this is because we enriched those loci that were bound by the protein or the histone modification of interest).

The **main task** in the **analysis of ChIP-seq data** is to **detect such peaks across the genome**. This can be achieved by detecting genomic regions that have **enrichment of aligned sequence reads** when compared against the **background**. For this purpose, we could, for example, use a proper **statistical test**. However, detecting peaks is **not a trivial task** as a **few issues and biases need to be addressed**. For example:

- It has been shown that **ChIP-seq data suffers** from **genomic regional biases** that is introduced by e.g., **mapping biases**, **chromatin structure** and **genome copy number variation** (Zhang et. al, 2008).

- When we perform **paired-end sequencing**, **we get two peaks** for each of the read pairs that **correspond to one TF binding event**. Here, we need to **shift the two peaks towards the 3´ end of the DNA fragment** so that we end up with only `1` peak (see Figure 5).
- If we use a statistical test to detect peaks across the entire genome, we need to address the **multiple hypothesis testing issue** (that we learned about in **chapter 7**).

Figure

Figure 4. Schematic illustration of ChIP-seq assay for protein-DNA complex (second row, left) and histone modification (second row, right). The bottom section shows preparation for sequencing via different sequencing technologies. The Illumina technology is more familiar to us as we learned about it in **chapter 5**.

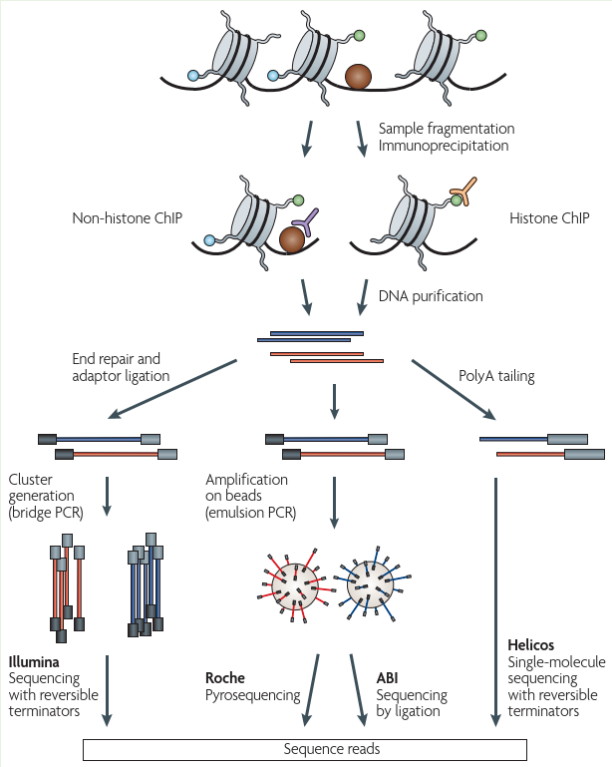


Figure source

Note

Here, we do not cover more details about **peak detection** (aka. **feature detection**).

Bioinformatics tools such as MACS address many of the issues and to reliably detect peaks. The **loci** marked by the **peaks** show **where in the genome** of a **particular cell type**, a **protein (or TF) of interest** had been **bound** or a particular **histone modification of interest** has been observed.

Figure

Figure 5. Schematic illustration of a peak from ChIP-seq data and the procedure to shift peaks from read pairs to form one peak.

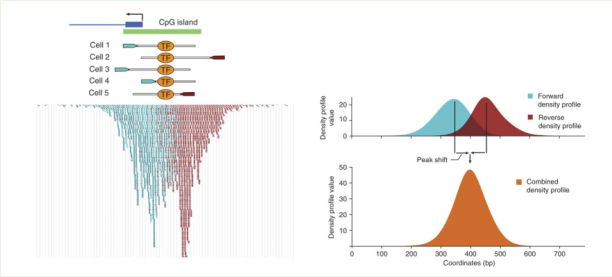


Figure source

Info

BED file format

A **file format** usually used to store ChIP-seq (or ATAC-seq) experiment peak coordinates is the *browser extensible data* (**BED**) file format. This file format is also widely used for **other purposes**.

BED is a **tab-delimited text file** that is used to store the genomic coordinates of features (in the first `3` required fields/columns) such as a ChIP-seq peaks (one feature per line). The `9` remaining optional fields/columns add more information about each feature such as giving it a name, a score, or information about the DNA strand on which the feature is located.

For example, the following excerpt shows the content of the first `5` fields of a BED file resulted from detecting peaks from an ATAC-seq experiment on a VCaP cell line sample using the MACS tool.

```
chr1    181132  181846  VCaP_peak_1     162
chr1    591175  591421  VCaP_peak_2     38
chr1    778367  779742  VCaP_peak_3     768
chr1    812056  812402  VCaP_peak_4     92
```

Note: **bigBed** file format is another file format based on the BED file format which is an **indexed binary file format**. Thanks to the indexing, it is much faster to access a particular feature (or set of features) than the regular BED files which makes it suitable for storing large amount of data.

To learn more about the BED file format, please read its Wikipedia entry here.

Task

❶ Can we use **ChIP-seq** to study where **RNA polymerase** binds? Please justify your answer. If your answer is yes, what can the data obtained from such experiment tells us?

❷ Can we use **ChIP-seq** to study a **TF** against which there is **no antibody available**?

ChIP-seq data databases

Over the years, **thousands of ChIP-seq experiments** have been performed for **different TFs** and **different cell types**. To keep track of these experiments, **specialized databases** have been developed that **collect** the data from these experiments, **process them uniformly** and **reliably**, and **make the results available in their databases**. [2] of such databases are Gene Transcription Regulation Database (GTRD) and UniBind.

Quote

GTRD

> The most complete collection of uniformly processed ChIP-seq data on identification of transcription factor binding sites for human and mouse. Convenient web interface with advanced search, browsing and genome browser based on the BioUML platform.

Source

Quote

UniBind

> **UniBind** is a comprehensive map of direct interactions between transcription factor (TFs) and DNA. High confidence TF binding site predictions were obtained from uniform processing of thousands of ChIP-seq data sets using the ChIP-eat software.

Source

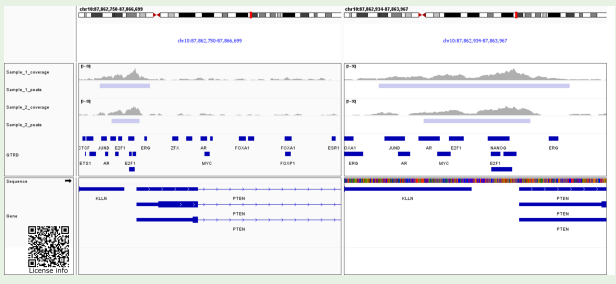
GTRD is a collection of **uniformly processed ChIP-seq data** used to identify **transcription factor binding sites** in the **human** genome as well as in the **mouse** genome. This database collects thousands of ChIP-seq experiments (e.g., `21988` experiments in *Homo sapiens* in version `21.12`) and data from thousands of unique TF.

The **data** from these databases can be used for **multiple purposes**. For example, **peaks** detected from an **ATAC-seq experiment** can be **annotated** to find out which TFs are known to bind to the open chromatin (i.e., accessible) genomic regions (see Figure 6). One bioinformatics tool that we could use for this purpose (i.e., the intersection between a set of ATAC peaks and a set of TF binding site coordinate from a database such as GTRD) is bedtools intersect. Bedtools is a **versatile bioinformatics tool** that can be used to perform ***genome arithmetic*** (i.e., set theory on the genome). For example, we can use this tool to find the **intersection**, **union**, and **difference** between **two sets of genomic features**.

Figure

Figure 6. IGV screenshot at the *PTEN* gene locus representing ATAC-seq data coverage and called peaks for `2` samples from prostate tissue. In addition, a subset of GTRD database data shows what transcription factors can potentially bind to the loci with detected accessible chromatin.

The right panel shows the same locus on the left panel but zoomed on the detected peaks loci.



Note how the length of the detected peaks differ across the two samples.

Other epigenetic modifications

In this chapter, we did not cover many other **epigenetic modifications** that are known to **regulate the gene expression** as well as the **assays** and **bioinformatics tools** that have developed to **detect and analyze such modifications**. Here, we only briefly name a few of them.

DNA methylation is an epigenetic mechanism that **typically silences gene expression**. Multiple assays such as bisulfite sequencing, reduced representation bisulfite sequencing (RRBS), and methylated DNA immunoprecipitation (MeDIP) have been developed to detect DNA methylation. Bismark is a bioinformatics tool for the analysis of bisulfite sequencing data and qsea is another bioinformatics tool for the analysis of MeDIP-seq data.

In this chapter, we learned about TFs and enhancers. According to Struhl (1999), in eukaryotes, **RNA polymerase II** requires the **TFs' help** to be able to **bind to promoters** in order to **transcribe mRNA**. Enhancers, on the other hand, **provide binding sites** for **regulatory proteins** such as **activator proteins** that attract TFs that help the RNA polymerase II to bind to the promoter region (**note** that it is also possible that **repressor proteins** bind to the enhancer region resulting in the **inhibition** of TF and RNA polymerase binding and transcription activity). This is achieved via the **formation of DNA loops** (see Figure 7). According to Jin et al. (2013), approximately `50` % of the human genes are involved in **long-range enhancer-promoter interaction**. Hi-C assay which is a chromosome conformation capture assay can be used to **detect and measure** such **DNA loops** among other use cases. For a list of available tools for Hi-C data analysis please check this GitHub repository.

Figure

Figure 7. Interaction of **enhancer** and **promoter** region via the formation of **DNA loops**.

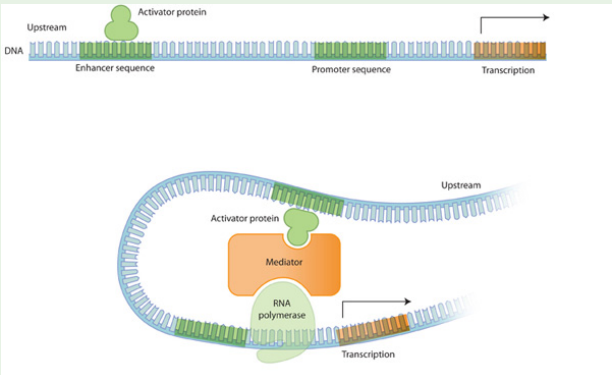


Figure source

A note on the integration of different data types

Each data type we introduced (and those that we did not introduce) in this chapter, reveals only a **partial picture** about the **regulation of gene expression**. One approach that lets us us to **form a more complete picture** about the regulation of gene expression is to **use** and to **analyze** these different data types **together** or in other words **to integrate different data types**. We already were introduced to a simple example of **data integration** when we **annotated** the **ATAC-seq peaks** with the a **subset of TF data from the GTRD database** (see. Figure 6). More **advanced approaches** as well as **bioinformatics tools** have been developed for **data integration**. For instance, ChromHMM can **integrate** data from **various histone modifications** obtained from **multiple ChIP-seq experiments** to **infer the chromatin state** (e.g., see figure 1 in ChromHMM article).

Task

Now, that we know data integration can provide us with a more complete picture, please go back to Figure 2 and look at the first ATAC-seq track, especially where we have ATAC peaks. For these regions and their vicinity, check the track with multiple ChIP-seq data (the bottommost track), and the gene track. Using these, can we tell apart different types of ATAC peaks (for example, whether they are promoter peaks or enhancer peaks)? What else can be inferred if we integrate these data?

ENCODE project

Now, that we have learned about multiple assays that are used to characterize transcriptome and epigenome (e.g., RNA-seq and ChIP-seq), we can briefly learn about the **ENCODE project**.

Info

ENCODE project

We have learned that only about `1`-`2`% of the human genome are **protein-coding**. What is the purpose/function of the remaining `98`-`99`%?

Encyclopedia of DNA Elements (ENCODE) project started in **2003** as a **follow-up** to the **Human Genome Project** (HGP) with the aim of **identifying all**

functional elements in the **human genome**. To achieve their aim, they used different assays such as **RNA-seq**, **ChIP-seq**, **DNase-seq**, etc.

ENCODE project had a major data release and publication in 2012. In 2020, the result of the **third phase** of the ENCODE project (ENCODE 3) was published.

Here are a **few results** from the ENCODE project:

- About `75` % the human genome is transcribed (based on results from `15` cell lines).
- About `57` % of the `75` % of the transcribed human geneome, an individual cell type undergoes transcription.
- About `8` % of the human genome shows protein binding (based on ChIP-seq data).
- About `15` % of the human genome shows open chromatin (based on DNase-seq data).

More information about the ENCODE project, its ongoing efforts, as well as tutorials can be found from here.

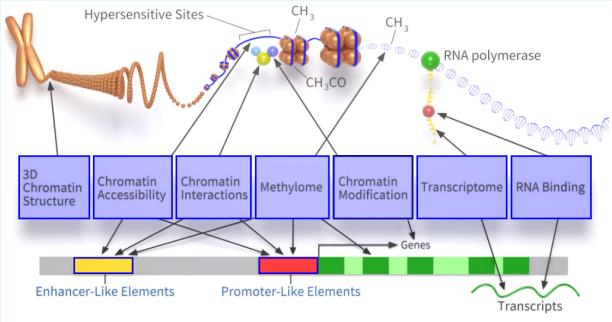


Figure source

⁂

Chapter 9. Proteome, Gene Ontology, biological pathways, and enrichment analysis

So far, we have learned about the **genome**, the **transcriptome**, and the **epigenome**. The genome encodes the genetic information needed to produce gene products. The transcriptome is the collection of all transcribed DNA sequences in a particular cell, cell type, or an organism. And, the epigenome is the collection of all epigenetic modifications that regulates the expression of gene products.

In this chapter, we proceed to learn how bioinformaticians study the **proteome** (i.e., **proteomics**). We define the proteome as *the collection of all proteins that are encoded by the genome and expressed in a particular cell*. Proteins are essential biomolecules and participate practically in every process within cells. In addition to the proteome, we also learn about, Gene Ontology and biological pathways as two key resources from which we can obtain information about the functional role of proteins. Finally, we learn about performing **enrichment analysis** as a useful approach to extract meaning from a list of genes or gene products.

Proteome and proteomics

Proteomics is the scientific study of the proteome at a large-scale. Proteomics is concerned with many topics involved with proteins. For example, we may be interested to know, **which proteins**, **in which cell type**, **where in the cell**, and **when** they are expressed and **by how much**. Proteomics is also concerned with the study of **modifications** that occur to proteins **after the translation stage** such as phosphorylation. These are known as **post-translational modifications**. The **interaction of proteins** with each other, as well as their **participation in biological pathways** are also the subject of study in proteomics. Understanding protein-protein interaction is important because many processes at the cellular level are governed by such interactions.

Multiple high-throughput techniques have been developed to study the proteome. Techniques based on mass spectrometry (MS) are commonly used in proteomics. The main idea behind MS is that different compounds can be uniquely identified by their mass and MS measures the masses of molecules such as peptides and proteins accurately. Thus, in proteomics, MS has been used to identify and quantify proteins.

Tip

You can learn about (or review) the MS technique by watching this 5-minute video made by the Khan Academy.

In early times of proteomics, proteins obtained from cell lysate were separated by two-dimensional gel electrophoresis (2-DE). This was, then, followed by mass spectrometry to identify the separated proteins. The throughput of this approach is limited due to the use of 2-DE step, and thus **gel-free approaches** have become more popular nowadays as they are less laborious than the 2-DE and help increase the throughput. LC-MS/MS (i.e., liquid-chromatography (LC) coupled to tandem mass spectrometry (MS/MS)) is a gel-free method that is popular in proteomics.

In the LC-MS/MS method, proteins are **extracted** from a sample and subsequently **digested** using, for example, the trypsin enzyme (or in other words, proteins undergo **proteolysis**). Next, **liquid-chromatography** is used to separate the resulting **peptides**. Subsequently, two rounds of MS are used to analyze the separated peptides (the **second round of MS** is used to improve the **sensitivity** and **selectivity**). This step produces **mass spectra** for the **peptides**. Next, a **pattern-matching algorithm** searches for these mass spectra in a **specialized database** and assigns each of them to a particular peptide. The reason peptides are used instead of whole proteins is due the higher sensitivity of identification methods (Aebersold and Mann, 2003).

Shotgun proteomics and **targeted proteomics** are two proteomics **strategies** that use the LC-MS/MS method. The use of different mass spectrometry methods, is what separates these two strategies apart. Shotgun proteomics is suitable for **protein discovery** enabling the discovery of a maximal number of proteins. However, it is **not suitable for quantification purposes** in case a large number of samples need to be quantified. On the other hand, targeted proteomics, can be used to **detect and quantify of a predefined set of proteins across many samples** in a **reproducible** manner. However, **a large part of the proteome remains undetected and unquantified** using the targeted proteomics strategy.

In order to quantify the **relative abundances** of the proteins in a sample via the LC-MS/MS method, proteins need to be **labeled** with **stable isotopes** (or in other words perform **isotopic-labeling**). This means that to be able to quantify all of the proteins in a sample, we need to synthesize different reagents capable of labeling different protein groups. The isotopic-labeling sets the throughput limit of LC-MS method to a few hundred peptides per analysis.

There are other mass spectrometry-based techniques used in quantitative proteomics. For example, we could name

- Isobaric tags for relative and absolute quantitation (iTRAQ),

- Stable Isotope Labeling by/with Amino acids in Cell culture (SILAC),
- Selected reaction monitoring (SRM), and
- Sequential windowed acquisition of all theoretical fragmentation - mass spectrometry (SWATH-MS). SWATH-MS is one of the state-of-the-art, label-free technique that has increased the throughput from a few hundreds to a few thousands (if interested, you can learn more about this method by reading this review article).

Are mRNA abundances representative of protein abundances?

In chapter 7, we learned about transcriptome and how to profile it e.g., with RNA-seq. We also know that the mRNA transcripts will be translated into proteins. We may suggest that, if we know the abundance of mRNA transcripts, we can reliably infer the abundance of their corresponding proteins without the need to measure the protein abundances. But as it turns out, this is not the case and the mRNA abundance may not be representative of protein abundances. For example, Latonen and colleagues (2018), while integrating the protein and mRNA abundance data obtained from prostate cancer tissue, observed that only around `73` % of the `3310` detected and quantified proteins showed positive correlation with the mRNA expression (see Figure 1). This can be partly explained by the dynamic nature of the proteome as it changes over time due to multiple processes such as post-translational modification. Thus, we may conclude that a complex biological system cannot be entirely and reliably characterized solely by gene expression profiling.

Figure

Figure 1. Protein levels may not be represented by mRNA levels.

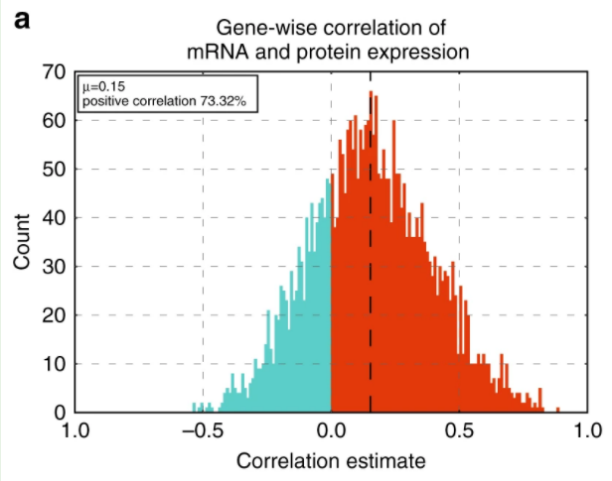


Figure source

Info

Similar to the Human Genome Project, there is a Human Proteome Project (HPP) that is coordinated by the Human Proteome Organization (HUPO). The aim of HPP is to

> revolutionize our understanding of the human proteome ... It is designed to map the entire human proteome in a systematic effort using currently available and emerging techniques. source

More-info

This interesting resource, shows the **proteomics' timeline** starting from `1897` and the **discovery of electrons** up until recent years.

Protein databases

We have already learned about UniProtKB database which stores protein sequences as well as other information about the known biological functions. There are other protein databases that store proteomics data such as:

- IntAct that stores information about protein-protein interactions.
- HuRI (The human Reference Interactome) that stores information about protein-protein interactions in humans.
- PRIDE (PRoteomics IDEntifications) stores experimental evidence of published protein and peptide identifications. In other words, this database is a repository for mass spectrometry data.
- InterPro InterPro provides functional analysis of proteins by classifying them into families and predicting domains and important sites.
- etc.

Ontology and biological pathway databases

Proteins are essential biomolecules in that they participate in practically every cellular activities and fulfill their functional roles. There are different resources to

learn about such functional roles. In this chapter, we use `2` key databases for this purpose, namely the Gene Ontology (GO) as an instance of ontology database and the Kyoto Encyclopedia of Genes and Genomes (KEGG) as an instance of biological pathway database.

Ontology and Gene Ontology (GO)

Let us start by the word *ontology* in the Gene Ontology (GO). Ontology has a few different meanings (e.g see here). For our purposes in this chapter, ontology refers to **a formal representation of a body of knowledge within a given domain**. An ontology has a set of **terms** or **concepts** that are **related to each other**. The *domain* in the GO, is the **biological domain**. Thus, GO has **a set of terms** that **represents our knowledge about biology in a formal manner**.

One thing that makes ontologies, such as the GO, **formal** is the use of **controlled vocabularies**. Controlled vocabularies **reduce the ambiguity that exists in normal human languages** where we can refer to the same concept with different names. For example, in our everyday language, we can refer to a tumor by calling it a *swelling*, a *lump*, or an *abnormal growth*. Controlled vocabularies, thus, provide a way to **organize knowledge** in a **consistent** manner that makes the **subsequent retrieval easier**.

In GO, a biological, controlled vocabulary is used **to describe and annotate genes or gene products across** `3` **aspects** of **molecular function**, **biological process**, and **cellular component**.

- **Molecular function** describes the **gene product activities** at the **molecular level**.
- **Biological process** describes the **processes** that result from **multiple molecular activities working together**.
- **Cellular component** specifies the **location within a cell** where a **gene product functions**. This could be either a **cellular compartments** such as mitochondrion, or **stable macromolecular complexes** such as the ribosome.

For example, let us check the GO annotation for the **β-globin** gene product encoded by the *HBB* gene (we have been using this gene in our earlier chapters' examples). The `3` aspects of the GO describe β-globin as:

- **Molecular function**: *oxygen carrier activity*, *oxygen binding*, ...
- **Biological process**: *oxygen transport*, *regulation of blood pressure*.
- **Cellular component**: *hemoglobin complex*, *extracellular region*, ...

Note that each of these aspects can be considered to be an ontology. Thus, we could say that the GO is composed of `3` ontologies.

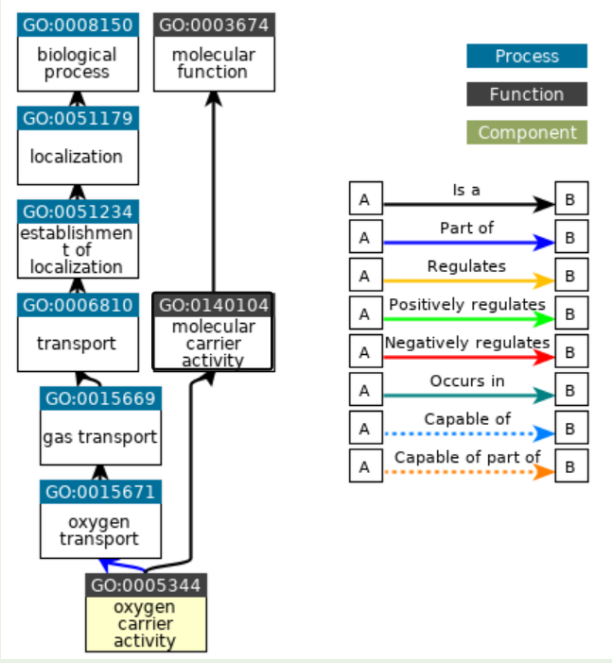
A **GO term** (aka. **GO class**) has **several parts**. Let us learn about them by inspecting the *oxygen carrier activity* term:

- A human-readable **term name** (e.g., *oxygen carrier activity*),
- A **GO ID** that is a unique `7`-digit identifier with `GO:` prefix (e.g., `GO:0005344`),
- An **aspect** that is one of the **three aspects** mentioned above (e.g., *molecular function*)
- A **definition** that describes the GO term (e.g., *binding to oxygen and delivering it to an acceptor molecule or a specific location.*)
- **Relationship to other GO terms** that represents how GO terms are related to each other
- Read here to learn about other **optional parts**.

We could use a graph to show GO terms and their relations to each other (see Figure 2). In the graph, **nodes** represent **GO terms** and the **edges** between the nodes represent the **relationship between the GO terms**. A few examples of relationship are: *is a*, *part of*, *regulates*, *negatively regulates*. For example, *oxygen carrier activity* (a **molecular function**) **is part of** *oxygen transport* (a **biological process**). It is also is a subtype of *molecular carrier activity* (a **molecular function**; see Figure 2). In the graph representation, we can have parents and children. A **parent** refers to the **node closer to the root(s)** of the graph, and a **child** refers to the **node closer to the leaf nodes**. A **child GO term** is **more specialized** than its **parent's GO term** (or parents' GO terms).

Figure

Figure 2. A graph representation of the *oxygen carrier activity* GO term and its relationship with other GO terms.



Note that the **arrowhead** indicates the direction of the relationship.

Figure source

Task

In Figure 2, which one of the *oxygen carrier activity* and *molecular carrier activity* is the **child** and which one the **parent**?

Based on your answer to the above, in Figure 2, is the *root* closer to the **top** of the figure or the **bottom** of the figure?

In addition to using graphs to represent the relationship among GO terms, a **tree structure** can also be used. For example, Figure 3 represent similar information as in Figure 2 but in a tree structure format.

Figure

Figure 3. A tree representation of the *oxygen carrier activity* GO term and its relationship with other GO terms.

Figure source

Task

Inspect Figures 2 and 3 side-by-side. To which aspect (i.e., *biological process*, *molecular function*, or *cellular component*) do the terms starting by the Ⓟ symbol in the tree structure refer?

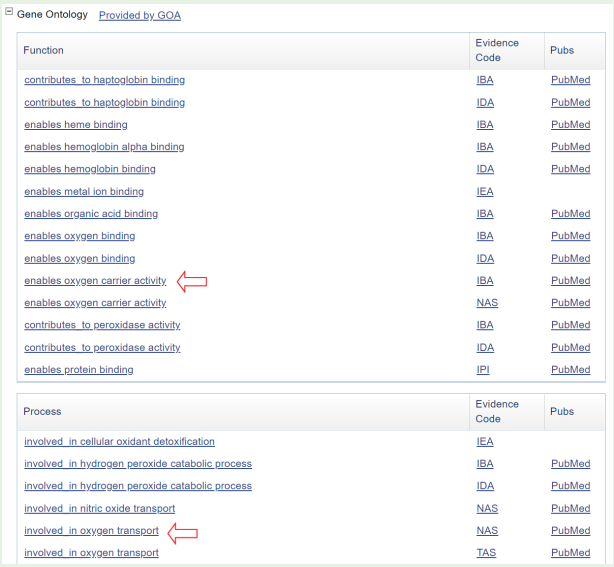
Tip

Biological databases such as **NCBI Gene** and **UniProt** provide us with GO annotations for our genes and gene products of interest (e.g., see Figure 4).

Figure

Figure 4. GO annotation for HBB can be obtained from biological databases such as (A) *NCBI Gene* and (B) *UniProt*.

A. NCBI Gene entry for *HBB*.

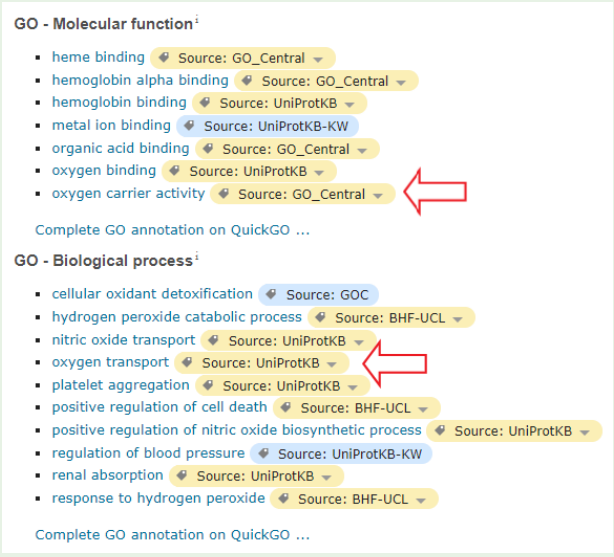


Gene Ontology Provided by GOA

Function	Evidence Code	Pubs
contributes_to haptoglobin binding	IBA	PubMed
contributes_to haptoglobin binding	IDA	PubMed
enables heme binding	IBA	PubMed
enables hemoglobin alpha binding	IBA	PubMed
enables hemoglobin binding	IDA	PubMed
enables metal ion binding	IEA	
enables organic acid binding	IBA	PubMed
enables oxygen binding	IBA	PubMed
enables oxygen binding	IDA	PubMed
enables oxygen carrier activity	IBA	PubMed
enables oxygen carrier activity	NAS	PubMed
contributes_to peroxidase activity	IBA	PubMed
contributes_to peroxidase activity	IDA	PubMed
enables protein binding	IPI	PubMed

Process	Evidence Code	Pubs
involved_in cellular oxidant detoxification	IEA	
involved_in hydrogen peroxide catabolic process	IBA	PubMed
involved_in hydrogen peroxide catabolic process	IDA	PubMed
involved_in nitric oxide transport	NAS	PubMed
involved_in oxygen transport	NAS	PubMed
involved_in oxygen transport	TAS	PubMed

B. UniProtKB entry for HBB.



GO - Molecular function

- heme binding — Source: GO_Central
- hemoglobin alpha binding — Source: GO_Central
- hemoglobin binding — Source: UniProtKB
- metal ion binding — Source: UniProtKB-KW
- organic acid binding — Source: GO_Central
- oxygen binding — Source: UniProtKB
- oxygen carrier activity — Source: GO_Central

Complete GO annotation on QuickGO ...

GO - Biological process

- cellular oxidant detoxification — Source: GOC
- hydrogen peroxide catabolic process — Source: BHF-UCL
- nitric oxide transport — Source: UniProtKB
- oxygen transport — Source: UniProtKB
- platelet aggregation — Source: UniProtKB
- positive regulation of cell death — Source: BHF-UCL
- positive regulation of nitric oxide biosynthetic process — Source: UniProtKB
- regulation of blood pressure — Source: UniProtKB-KW
- renal absorption — Source: UniProtKB
- response to hydrogen peroxide — Source: BHF-UCL

Complete GO annotation on QuickGO ...

As we can see from Figure 4 A, there is a column called ***Evidence Code***.
These codes represent the **annotation source** which could be a scientific publication or computationally inferred annotation e.g., based on sequence similarity. Here, we list a few of them. The more complete list of *Evidence Codes* and their meanings can be accessed from this page.

- Inferred from Biological aspect of Ancestor (IBA)

- Inferred from Direct Assay (IDA)
- Traceable Author Statement (TAS)
- Non-traceable Author Statement (NAS)

Quote

About Gene Ontology

> The mission of the GO Consortium is to develop a comprehensive, **computational model of biological systems**, ranging from the molecular to the organism level, across the multiplicity of species in the tree of life.
> The Gene Ontology (GO) knowledgebase is the world's largest source of information on the functions of genes. This knowledge is both human-readable and machine-readable, and is a foundation for computational analysis of large-scale molecular biology and genetics experiments in biomedical research. source

> GO aims to represent the current state of knowledge in biology, hence it is constantly revised and expanded as biological knowledge accumulates. Changes are made on a weekly basis (most relatively minor). source

More-info

There are other ontologies that we did not discuss here. For example, we can name:

- Sequence Ontology (SO) with the aim of defining sequence features used in biological sequence annotation.
- Phenotype and Trait Ontology (PATO) which is an ontology for phenotypic qualities (properties, attributes or characteristics).

A large list of available ontologies can be found here.

Browsing the GO with AmiGO

We can use AmiGo to browse the GO. Watch this 2-minute video (**note** that it is a silent video) to see how we can use AmiGo to browse the GO and to get to the *oxygen carrier activity* GO term (`GO:0005344`) that we have been using in this chapter. In this video, we start from the AmiGO's first page. We start browsing the GO by pressing the *Go* button under the *Browse the Ontology* box. Next, we limit our

results to *Homo sapiens* by introducing a filter over the *Organism*. Note the decrease in the number of annotations after the introduction of this filter. Also, look how we traverse the tree starting from the *molecular function* by pressing on the ✚ symbols until we reach the GO term of interest. In the end, by clicking on the GO term (*oxygen carrier activity* here), a pop up window appears that gives us more information about that particular GO term. Clicking on the link next to the *Term* line takes us to the GO term dedicated web page (look at the URL in the web browser and you can see the GO ID associated to the*oxygen carrier activity* term in the URL). The GO ID is also the same for the *Accession*. If we scroll through this page, we can see the gene products related to this GO term. Again, we can introduce filters as we do here by choosing only entries related to *Homo sapiens*. We can see that HBB is listed as a gene product under this term as we expected. We can also see *Evidence Codes* that we learned above for each of the entries.

Searching the GO with AmiGo

We can use AmiGo, also, to search for the *annotation*, *ontology*, and *genes and gene products*. Using the search, we can reach the page for the *oxygen carrier activity* in a fewer steps as we needed above starting by searching the *Ontology*. Watch this 1-minute, silent video to see how.

Let us also start searching for the *oxygen carrier activity* in Homo sapiens starting from the *Annotation*. We would expect to get to the `15` entries that we got when we browsed the GO with AmiGO. This may need paying attention to a few details. See this 2-minute, silent video to see how (⚠ **Note**. Due to the GO updates, there are currently `2` more entries than can be seen in the videos).

Biological pathways and Kyoto Encyclopedia of Genes and Genomes (KEGG)

In the previous section, we learned that when **multiple molecular activities** work together, they form a **biological process**. The GO documentation, however, differentiates between a *biological process* and *biological pathway* by stating that these are not equivalent. According to the GO documentation, full description of a pathway requires further representation of the dynamics and dependencies that exists among molecules involved in a pathway. We may ask, what is a **biological pathway** then? Collins online dictionary defines a *pathway* in biochemistry as *a sequence of reactions, usually controlled and catalyzed by enzymes, by which one organic substance is converted to another*. Additionally, NHGRI defines a biological pathway as *a series of actions among molecules in a cell that leads to a certain product or a change in the cell. It can trigger the assembly of new molecules, such as a fat or protein, turn genes on and off, or spur a cell to move*.

There are different types of biological pathways such as:

- **Gene regulatory pathways** that regulate the expression of genes (e.g., turning genes on and off)
- **Metabolic pathways** such as:
  - Catabolic pathways that breaks down complex molecules (e.g., those obtained from food) and release energy that can be stored in energy-carrying molecules for later use.
  - Anabolic pathways that synthesize larger, more complex molecules from smaller molecules (e.g., fatty acid synthesis).
- **Signaling** (or **signal transduction**) **pathways** that transmit chemical or physical signals from outside of a cell to its interior via a series of molecular events such as protein phosphorylation resulting in a cellular response (e.g., cellular motion).

We may ask what are the benefits of using biological pathways, for example, in the field of bioinformatics? One possible benefit is that we could use the information encoded in pathways to help us getting clues about, for example, a disease. For instance, in the context of a particular disease, we could compare two groups of healthy and diseased participants and check whether we can identify any biological pathway that is different across these two groups (or in other words, ask, can we find one or more biological pathways that are disrupted in the diseased participants when compared to the healthy group?). We will see an example, later in this chapter, when we learn about **enrichment analysis**.

Now, that we know about biological pathways and one possible benefit, we can ask from where can we access information about them? As you may have guessed already, there are specialized databases that store information about biological pathways. One such database is the Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway database. KEGG pathway database is a **human-curated** resource that collects a large collection of signaling, metabolic and gene regulatory pathways. Using this resource, we can search for a pathway, or browse the list of available pathways listed at the bottom of KEGG pathway web page.

Let us look at a KEGG pathway example, namely the JAK-STAT signaling pathway. This pathway receives chemical signals from outside of a cell and transmits it to the cell nucleus which results in certain genes to be transcribed/expressed (see Figure 5). In this figure,

- Green boxes denote gene products, mostly proteins but they may also denote RNA.
- Lines denote the connection between different gene products.
- `2` parallel lines denote the cell surface. We can also see a receptor on the cell surface.
- Vertical dashed line denote the boundary between the nucleus and the cytoplasm.

- O denotes chemical compounds, DNA, and other molecule. In this figure, O denotes DNA regulated by transcription factors.
- Stadiums (i.e., round-cornered boxes) denote relationships to other pathways. For example, *MAPK signaling pathway* and *Apoptosis* are `2` of multiple pathways listed in this figure.

Here, we explained the meaning of a few of the symbols. To learn more, please read the help page on KEGG pathways (see also Figure 6). This help page provides more information other than the meanings of the symbols. For example, it also explains the color codes used in pathways.

Note that we can click on the boxes in a KEGG pathway and by doing so we are redirected to an entry in the KEGG genes/proteins database. Let us click on the **IRF9** box. In the page that opens, on the *Entry* line, we can see a number i.e., `10379`. This is an *Entrez ID*. For example, we can search the *NCBI Gene* database for this ID. We can see that the search result points to IRF9. In a KEGG pathway, we can also click on the stadiums. This takes us to the pathway the stadium is representing. For example, let us click on the stadium denoting *MAPK signaling pathway*. You can watch this 2-minute, silent video that shows searching for a pathway (i.e., JAK-STAT signaling pathway) and interacting with this pathway as described in this paragraph. In this video, we use the `hsa` prefix that denotes Homo sapiens when performing the search (the default `map` prefix denotes the *reference pathway*).

Figure

Figure 5. JAK-STAT signaling pathway in the KEGG pathway database.

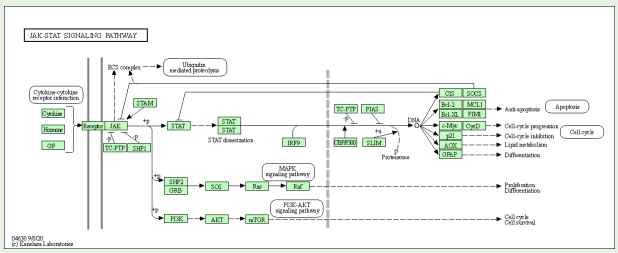


Figure source

Figure

Figure 6. KEGG pathway symbols and their meanings.

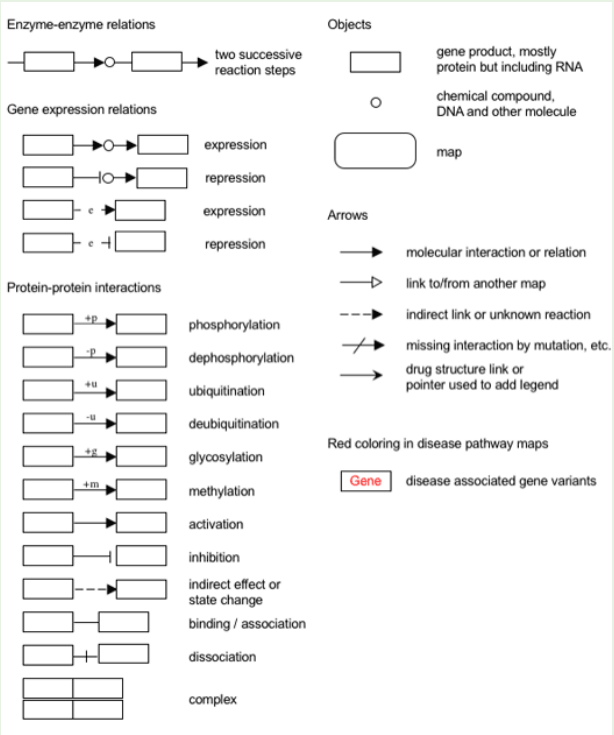


Figure source

Info

There are other pathway databases, such as:

- WikiPathways which is a community curated pathway database.
- Reactome which stores manually curated information about proteins' roles in biological pathways.
- PANTHER pathways which consists of over `177` biological pathways that are primarily signaling pathways. PANTHER is short for *Protein ANalysis THrough Evolutionary Relationships*.
- Pathway commons which aims to collect and disseminate biological pathway and interaction data from public pathway and interaction databases.

We should note that **many biological pathways cooperate to accomplish different tasks**. In that sense, there may not be real boundaries that separate pathways from

each other (this, somehow, could be observed from Figure 5 that certain cell responses in the JAK-STAT signaling pathway propagate to other pathways). The interaction among multiple **biological pathways** results in what is known as a **biological network**.

Enrichment analysis

In chapter 7, we learned about differential expression analysis which results in list of differentially expressed (DE) genes. What could we use this list of genes for? Is there **a way to extract meaning from this list of genes**, which often can be a long one, instead of inspecting them one at a time? Is there an approach to break down this long list into smaller sets of DE genes that are related to each other with some respect? For example, an approach that takes the long list of DE genes and identify groups of genes that function under the same pathways or related to a biological process.

One way is to perform *enrichment analysis*. For example, we could check **whether our list has more genes than one would expect to occur just by chance** that are **related to a particular biological pathway** or in other words, **whether our list is enriched with a set of genes related to a particular biological pathway** (instead of biological pathways, we could also use **Gene Ontology annotations** or **any other annotations**). Note that, here, we used the list of DE genes as an example, but we could perform enrichment analysis over a list of genes/features that is obtained from whatever bioinformatics analysis. For example, imagine that we have obtained a set of accessible chromatin regions by analyzing ATAC-seq data, and we want to know whether e.g., a particular chromosome has more open chromatin regions than expected. To answer to this question, we could perform enrichment analysis. A benefit of enrichment analysis, for example in the case of DE gene list, is that we can turn a gene list to a list of biological features. It is much easier to interpret the latter list than the former.

To perform enrichment analysis, we can use a proper **statistical test** such as the **Fisher's exact test**. Let us watch a 5-minute video about *Fisher's exact test* and its use in enrichment analysis.

To perform Fisher's exact test, we need to know the following counts (i.e., N, M, n, and k):

- A **background set** of N items that contains **all possible genes we could have** on our gene list (e.g., list of all known genes in humans).
- M out of those N genes have feature *X* (e.g., related to biological pathway *X*).
- We have the gene list *A* that contains n genes out of N obtained as a result of some analysis such as differential expression analysis from the background set.

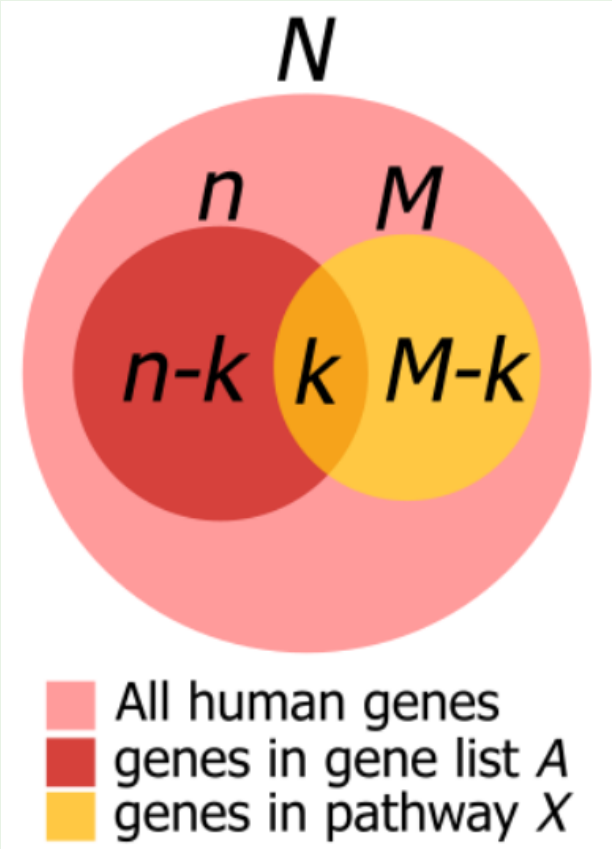
- k out of the n genes have feature *X* (e.g., related to biological pathway *X*; see Figure 7).

Next, we follow the following steps to test for enrichment:

- Make the **null hypothesis** that the k genes having feature *X* arise **due to chance**.
- Calculate a **p-value** to quantify **how often** we would **get k or more genes having feature *X*** if **gene list *A* was a random selection of n genes from the background set**.
- Choose features (e.g., biological pathways) with p-values **smaller than** a **significance threshold** as interesting.

Figure

Figure 7. An example of counts involved in calculating Fisher's exact test.



Let us consider the example shown in Table 1. We ask: Is $10/260$ gene list annotations a lot when compared against the whole background rate of $210/20000$ given the significance threshold set at `0.01`? Fisher's exact test says **yes** to this question since the calculated p-value `0.000442` is smaller than significance threshold.

Table

Table 1. Counts obtained for a gene list *A* and a feature *X*.

	Have feature ***X***	Does not have feature ***X***	Row total
On the gene list	**10**	250	**260**
Not on the gene list	200	19540	19740
Column total	**210**	19790	**20000**

Example

Performing Fisher's exact test over data in Table 1 using **Python**.

```
from scipy.stats import fisher_exact
table = np.array([[10, 250], [200, 19540]])
odds_ratio, p_value = fisher_exact(table, alternative='two-sided')
print('P-value: %.6f' %p_value)
print('Odds-ratio: %.2f' %odds_ratio)
# Output: P-value: 0.000442
# Output: Odds-ratio: 3.91
```

Task

Inspect Table 1 and find the values that correspond to N, M, n, and k described above.

There is **one familiar issue** that we have to be aware of and address. In our example of biological pathway enrichment analysis, there are **many biological pathways that can be tested**. This means that we would perform hundreds of such statistical tests. This reminds us of the issue of **multiple hypothesis testing** we initially discussed about in earlier chapters. Therefore, we need to perform **multiple hypothesis testing correction**, if we want our results to be reliable. Remember that using p-value threshold of `0.05` and testing `100` pathways for enrichment means that `5` out of these `100` pathways would show up as enriched even if we have **constructed our list of DE genes entirely randomly**. Instead of a p-value **significance threshold**, it is often more useful to consider what is known as *False Discovery Rate* (FDR). FDR is the **fraction of positive results (i.e., discoveries) that are false**.

Benjamini-Hochberg procedure is a method for multiple hypothesis testing correction. P-values corrected (i.e., adjusted) by the Benjamini-Hochberg method can be **considered to be on the FDR scale**, so that setting a significance threshold of `0.05` means that `5`**% of pathways tested positive for enrichment** are expected to be **false positives** and **not** `5`**% of the pathways that we tested**.

Note that the enrichment analysis approach that we described here has a few limitations. To name one, this approach **only considers the gene counts in the lists** and it does not take into account other information that may be available and useful for our purposes. For example, it does not take into account this hypothetical situation where $gene_a$ is expressed `16` times more in $condition_1$ than in $condition_2$ while $gene_b$ is only expressed `4` times more in $condition_1$ than in $condition_2$. In this approach, **both of these genes are treated equally**. There are other approaches that try to use such additional information when extracting meaning from a list of genes. To learn more about other approaches, please read this review article by Khatri, Sirota, and Butte (2012).

The Database for Annotation, Visualization and Integrated Discovery (DAVID)

DAVID is an online bioinformatics tool that we can use to perform enrichment analysis. It uses a modified version of the Fisher's exact test p-value known as EASE score. We can upload a gene list that we have obtained from an analysis (e.g., differential gene expression analysis) and the tool can perform the enrichment analysis for different sets of annotations such as GO or KEGG pathways. DAVID can perform other types of analyses as we will see in the following video.

📹 This 13-minute video demonstrates how we can use the DAVID in practice.

To learn more about DAVID, you may read this help page provided by DAVID.

> **Note**
>
> Biological databases, like Gene Ontology, update frequently (e.g., weekly in the case of Gene Ontology). Consequently, as DAVID relies on these databases for annotation, its results may change post-update.

Further reading

- Mass spectrometry-based proteomics (Aebersold and Mann, 2003) available via Tampere University library here.
- Gene Ontology overview

- Primer on *Fisher's exact test* by Pathway commons

⁂

Chapter 10. Single-cell sequencing

So far, we have learned about the different high-throughput sequencing approaches used to study the genome (whole-genome sequencing (WGS) or DNA-seq), the transcriptome (RNA-seq), and the epigenome (ChIP-seq and ATAC-seq). One thing that these approaches have had in common is that they require a bulk of cells ranging from thousands to millions. As a result of this, the data we get from these approaches is **an average of the bulk of cells** which may be under **different conditions** and **states**, and of **different cell types**. We can think of the bulk sequencing methods using the **fruit smoothie** analogy (Chow, E., 2020). In a fruit smoothie, different fruits with different proportions are mixed up together contributing to the final result.

Single-cell sequencing (SCS) approaches enable us to **characterize the cells** in a sample **separately** and provides us with a **view of the heterogeneity** (or variability) that may exist across the **cell sub-populations** in a sample. This allows us to compare **cells and cell sub-populations** against each other.

Task

Think about the usefulness of the information that we get by performing SCS on a cell culture (where the cells are supposed to be more or less similar) and a clinical tissue such as a tumor sample. Which one does provide more information?

In 2009, the first ever SCS study, characterized the **transcriptome** of **one single cell** by performing RNA-seq (Tang et al., 2009). The first **single-cell DNA sequencing** experiment in human cancer cells was conducted in 2011, and the first **single-cell exome sequencing** experiment was conducted in 2012. However, thanks to recent single-cell technological developments as depicted in **Figure 1**, the process is becoming less manual, laborious, costly, and time-consuming and thus **more and more cells** are being **characterized** in each study. The increase in the number of cells has demanded for the development of novel data analysis techniques.

Figure

Figure 1. Development of single-cell technologies over that past two decades

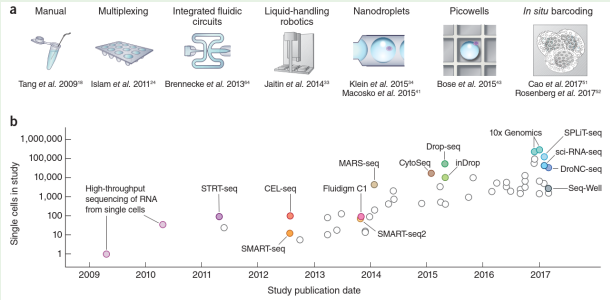


Figure 1 | Scaling of scRNA-seq experiments. (**a**) Key technologies that have allowed jumps in experimental scale. A jump to ~100 cells was enabled by sample multiplexing, and then a jump to ~1,000 cells was achieved by large-scale studies using integrated fluidic circuits, followed by a jump to several thousands of cells with liquid-handling robotics. Further orders-of-magnitude increases bringing the number of cells assayed into the tens of thousands were enabled by random capture technologies using nanodroplets and picowell technologies. Recent studies have used *in situ* barcoding to inexpensively reach the next order of magnitude of hundreds of thousands of cells. (**b**) Cell numbers reported in representative publications by publication date. Key technologies are indicated. A full table with corresponding numbers is available as **Supplementary Table 1**.

Figure source

Even though the early single-cell studies have focused on **characterizing the transcriptome** at the single-cell level, such as using **single-cell RNA-seq** (**scRNA-seq**), which is currently **the most common protocol**, other protocols such as **single-cell ATAC-seq** (**scATAC-seq**) and **single-cell ChIP-seq** (**scChIP-seq**) have been developed. scATAC-seq and scChIP-seq allow us to characterize the **chromatin accessibility** and **histone modifications** respectively and thus enable us to study the **gene regulation** at the single-cell level (see **Figure 2**).

Figure

Figure 2. Different single-cell protocols.

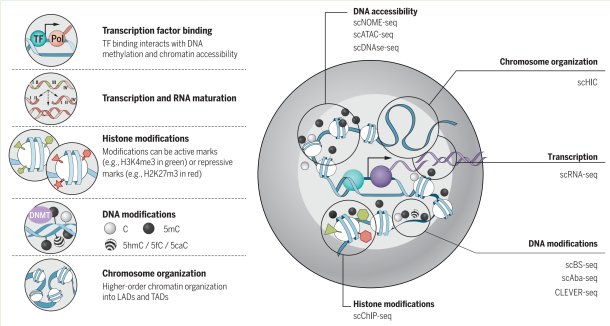


Fig. 1. Single-cell methods and heterogeneity of different molecular layers. (**Left**) Overview of different molecular layers that can be assayed using single-cell protocols. (**Right**) A cell with different layers of multi-omics measurements, as defined on the left. Concordance or heterogeneity respectively may exist between the different layers, and this can be recorded by single-cell sequencing and computationally evaluated.

Figure source

What makes applying the SCS to multiple cells at once possible is the clever use of **labelling techniques** (see **Figure 1a**) that allows us to **track the cell of origin** of the **DNA/RNA fragments** in our experiment and thus assign the corresponding sequencing reads to the cell they have been originated from.

There are currently a few ways and protocols to perform SCS. Here, we get to see one of these as described by Klein and colleagues (2015). This is a droplet microfluidic-based SCS method and will convey the main ideas used in single-cell sequencing. As it can be seen from Figure 1, this method can be used to study **tens of thousands of cells** in a single run.

As it can be seen from **Figure 3**, we start by **cells** that are **disassociated** from each other. Additionally, there are spherical objects called **hydrogels** that contain **primers** that are composed of **different sections**. All the primers in one hydrogel have an **identical *cell barcode***. **Cell barcode** is just **a sequence of nucleotides** that are used for **labelling** a **droplet** and eventually the **cell contained in that droplet**. In addition to the cell barcode, each primer contains **unique molecular identifier** (**UMI**), **sequencing adapters** necessary for sequencing, **promoter sequence** for the T7 RNAP and a few other things.

The next step in this method aims to put/encapsulate **only one cell** and **one hydrogel** in **one** very small drop of liquid, at the **nanoliter scale**, called **droplet**.

> **Note**
>
> Even though, the aim is to put one cell in each droplet, sometimes **no cell** or **more than one cell** enter **one droplet**.

The droplet contains substances that are used to **break down the cell membrane** (**lysis**) and the **reagents** needed for the **reactions** to go on. The **primers** are released once the **droplets** are **exposed** to the **ultraviolet (UV) light**. After lysis, the cell content interacts with the released primers which **in this method** starts a **reverse transcription reaction** inside the droplet resulting in **cDNA**. This is the starting point of the **library construction**. Once enough time has elapsed for the reactions to take place, the **droplets** can be **broken down**. Next, the droplets' content are **linearly amplified** resulting in a **sample** that includes **DNA fragments from different cells**. However, thanks to the **cell barcodes**, it is possible **to distinguish fragments originating from different cells**. This content undergoes **library preparation** before sequencing. After library preparation, we **perform sequencing** the same way as we learned about in chapter 4 (i.e., no special treatment is needed). Once we have the **sequencing reads**, we can use them for **data analysis**.

This forms the fundamental idea in SCS. Different protocols adhere to this fundamental idea, yet they diverge due to different details that they employ to achieve the same goal. These adaptations aim to enhance factors such as cost, effectiveness, and efficiency by enhancing data quality, quantity, and completeness which enable increasing the number of cells that can be used for downstream analysis and the number of detectable features per cell.

Example

An alternative single-cell barcoding technology based on **combinational indexing** overcomes the limitations of the relatively **high cost of isolating a single cell for bead-based barcoding technologies**. Combinational indexing-based barcoding technologies recognize single cells by the **addition of cellular barcodes in multiple rounds without isolating single cells**; these technologies include *Sci-Seq*, *Microwell-Seq* and *Split-Seq* (Lei et al., 2021).

Figure

Figure 3. Klein et al. (2015)'s approach to barcode thousands of cells.

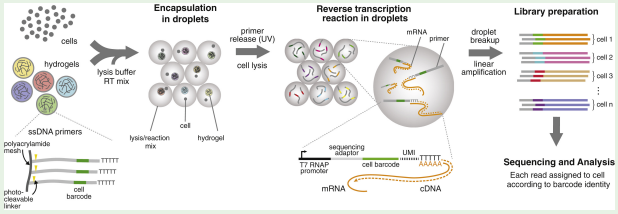


Figure source

Task

Look at the primers in **Figure 3**. Based on what you see, can you tell whether it is meant for any RNA species or only messenger RNA? Why (**Hint**. Refer to the *Transcriptome and gene expression profiling* chapter)?

Note

Currently, scRNA-seq is widely employed to characterize the **transcriptomes** of individual cells. Notably, two commonly utilized **platforms** are the droplet-based

10X Genomics Chromium and the plate-based Switching Mechanism at the 5′ End of RNA Template sequencing (**SMART-Seq**)

- The 10X Genomics platform is capable of isolating, labeling, amplifying and preparing a cDNA library from 5000 to 10,000 single cells at a high speed. However, it is biased to detect only the 3′ or 5′ end of the transcript. Additionally, it requires abundant cells in a single sample (recommended over 90%) are needed. Thus, it is not suitable for detecting rare samples containing few cells.
- SMART-Seq facilitates the detection of full-length transcripts. It is widely utilized in cancer research.

(**See**. Lei et al., 2021 for more information).

Note

The more cells we can characterize, the more the chance to characterize rare sub-populations in our sample.

Single-cell data preprocessing and data analysis

In this section, we learn about the general steps that are involved in single-cell data preprocessing and analysis.

As the **first step**, we can **separate** all the **sequencing reads** based on their **cell barcodes** and **assign reads** with **similar barcodes** to **one sample**. In other words, each sample will contain sequencing reads from a particular cell. Once this step is done, each cell barcode needs to be removed from the sequencing read as this was a nucleotide sequence that was artificially introduced to the end of DNA fragment and if it is not removed, it introduces a problem in the alignment step.

Note

During the preprocessing phase, the use of the **UMI**s enables the detection of **duplicate data** within our results. Specifically, if two DNA sequences possess similar UMIs, it becomes evident that a single fragment has been sequenced twice, leading to the presence of duplicate sequences.

The subsequent step, akin to bulk data analysis, involves the alignment of reads to a reference genome.

Following the alignment, particularly in scRNA-seq scenarios, the process proceeds to read counting, akin to the approach utilized in bulk RNA-seq data analysis.

The subsequent step involves conducting quality control procedures. This step encompasses certain shared quality control measures with bulk data analysis, such as assessing base call qualities and evaluating the alignment efficacy to the reference genome. However, single-cell data requires additional, unique quality control steps. For instance, each cell's data must undergo separate quality control assessments. Cells exhibiting unusually low or high read counts, as well as an atypical number of detected genes, warrant removal from the analysis. Another distinctive quality control aspect involves evaluating the presence of mitochondrial DNA. A high proportion of data originating from mitochondrial DNA might signal droplet construction failure.

⚠ Attention

In scRNA-seq data analysis, **background noise** can impact the accuracy of results. This noise emerges when not all reads linked to a particular cell barcode originate from the encapsulated cell itself. Two factors contributing to background noise are **barcode swapping** and **cell-free ambient RNA**.

Cell-free ambient RNA constitutes a pool of mRNA molecules released into a cell suspension, often stemming from stressed or apoptotic cells. If this ambient RNA becomes incorporated into the droplets, gets barcoded, and subsequently amplified alongside a cell's native mRNA, the sequencing reads originating from these molecules contribute to the background noise in the dataset. As ambient RNA contamination can distort the true cellular expression profiles, computational approaches to reduce noise are essential for obtaining accurate insights from scRNA-seq data. Figure 4 shows how ambient RNA can be incorporated into droplets.

Figure

Figure 4. Incorporation of ambient RNA into droplets.

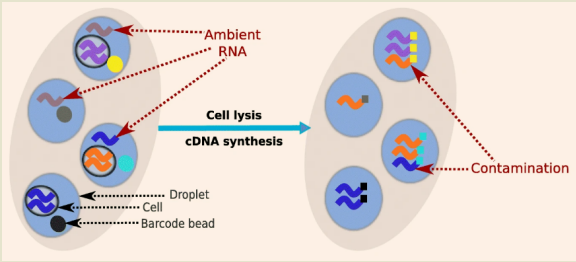


Figure source

Task

- What could potentially be responsible for an unusually low amount reads, associated with a cell in a single-cell sequencing experiment?
- What could potentially be responsible for an unusually high amount reads, associated with a cell in a single-cell sequencing experiment?

Answer

- A low quantity of reads might indicate a droplet failure. This could result from various factors, such as unsuccessful reactions.
- A high volume of reads could imply various scenarios, one of which might be the inclusion of multiple cells within a single droplet.

Note

One thing to keep in mind is that when we are preparing a tissue sample for SCS, we need to dissociate the cells as they are connected to each other via *cell adhesion molecules* such as *transmembrane adhesion proteins*. During the dissociation, the cells get stressed which affect the processes inside the cell. In the case of scRNA-seq, this can be seen e.g., in the expression patterns. To address this issue, we need to use some computational methods to filter out such effects so that we can study the original cell state that we were originally interested in studying and was the state before the cells were being dissociated.

Alternatively, the single-nucleus RNA sequencing (**snRNA-seq**) technique, which originated in 2013, can be employed. This technique aims to mitigate the introduction of artefactual transcriptional stress responses or transcriptional bias.

Note

In case we are consolidating data from **different experiments or laboratories**, ensuring a coherent and unified dataset might necessitate data correction procedures, such as **batch effect correction**.

To ensure comparability among data from distinct samples (i.e., cells here), we may need to perform normalization. The single-cell data normalization is slightly different from the bulk data normalization and it is a topic that is currently under ongoing development and different tools may offer different normalization techniques.

Following proper preprocessing and accounting for various factors that affect the data, in the case of scRNA-seq analysis, we end up with a *gene-cell matrix*. In this matrix, each row corresponds to a specific gene, while each column represents an individual cell.

Note

We may perform **feature selection** that involves choosing attributes that are informative or interesting to us. For instance, genes exhibiting uniform expression across various cells might lack substantial informative value. Thus, we may opt to examine only genes with notable variability, often quantified through variance.

With the data at hand, analysis of scRNA-seq data can start following a few basic steps. For instance, one of the first questions is about the similarity or dissimilarity among cells when compared to one another. We may ask: do these cells represent the same cell type but in different states, or whether the data suggests the existence of distinct cell types within our sample of cells?

Visual inspection can be employed as an initial approach to answer these questions. If only one gene were quantified for each cell, a *number line* could be employed to gauge the cell positions. For two quantified genes, the *coordinate plane* (aka the *Cartesian plane*) would be applicable to visualize cells in two dimensions. However, typical scRNA-seq experiments involve quantifying thousands of genes, which creates a need for visualization strategies that are more human-friendly.

Given that human perception readily accommodates up to three dimensions, the challenge arises: can we devise a visualization method for such high-dimensional data? Fortunately, algorithms have been developed to address this challenge. These algorithms engage in *dimensionality reduction*, transforming N-dimensional data into two or three dimensions for easy human comprehension. This process arranges cells on a 2D (or 3D) plane, positioning cells with similar gene expression profiles in closer

proximity. In essence, this dimensionality reduction process provides a means to explore and comprehend intricate scRNA-seq data structures, revealing relationships among cells and facilitating the identification of distinct cell types or states within the sample. Two such methods are:

- t-distributed stochastic neighbor embedding (tSNE). This algorithm primarily focuses on the **preservation of the local structure in the data**. It possesses a distinctive characteristic: it tries to emphasize the cluster separations such that it is easier for humans to tell the clusters apart and see the structure of the data. Consequently, this method is not entirely unbiased or neutral in nature. Hence, it is important to note that **the distances between clusters may not accurately reflect the actual distances**. For instance, if clusters A and B are positioned at opposite ends of a plot, it is not valid to deduce that these clusters are inherently more dissimilar than clusters A and C, especially when C is closer to A on the plot. Thus, we should exercise caution in avoiding overinterpretation of patterns observed in such visualizations.
- Uniform manifold approximation and projection (UMAP). This algorithm focuses on **preservation of both global and local structures in data** when performing dimensionality reduction. UMAP has gained popularity for its ability to strike a balance between global and local structure preservation.

✓ **Check**

Read this scientific article by Kobak and Linderman (2021) to find out how UMAP achieves its advantage over tSNE.

Important

Both tSNE and UMAP are **stochastic** in nature and are significantly impacted by the choice of **hyperparameters**. As a result, they may produce different results in different runs.

> **Definition**
>
> **Hyperparameter**. In this context and machine learning context, hyperparameter refers to a parameter whose value is set before the learning process begins and used to control the learning process. Additionally, its value is not learned from the data during training.

Task

What do you envision the appearance of a t-SNE plot for scRNA-seq data to be like when the data originates from a cell culture comprising only one cell type?

Once we have the visualization, we can use an additional algorithm, known as a *clustering algorithm*. This algorithm conducts clustering analysis to allocate cells into distinct clusters. It is crucial to fine-tune the algorithm's parameters appropriately to ensure that the clustering outcomes align with the inherent biological patterns. An example of such an algorithm is the **graph-based clustering** algorithm. This category of algorithms examines the vicinity of similar cells and leverages this information to construct clusters.

Note

There are techniques available for determining the optimal number of clusters based on the data itself. Nonetheless, it is crucial to be aware that the optimal cluster count might not faithfully capture the underlying biological factors.

Given that we have chosen parameter values for the clustering algorithm that align with our data, individual clusters could potentially signify distinct **cell types** (**subtypes** or even different **cellular states**). If we know that our data is composed of different cell types, we can try to allocate each cluster to a specific cell type—essentially **annotating the clusters**.

Check

For an overview and reflection of **what a cell type is**, please browse through this article by Fleck, Camp, and Treutlein (2023).

Let us recall from previous chapters that certain genes were expressed mainly in specific cell types, distinct from others. In the context of scRNA-seq, we can harness our understanding of gene expression specificity to ascertain the cell type corresponding to each cluster of cells. These genes, renowned for their specificity, are referred to as *cell-type-specific marker genes*, often simplified to *marker genes*. These genes might be familiar to us from prior research, or they can be identified via statistical methods. In the latter case, referred to as *marker identification*, we can pinpoint a set of genes with elevated expression. Subsequently, techniques such as **enrichment analysis**, previously explored in an earlier chapter, can be employed to discern the functions of these genes. This process, in turn, aids us in unraveling the

distinct cell type associated with each cluster through the characterization of the marker genes' roles.

Advanced scRNA-seq data analyses

Above, we outlined a few of the fundamental steps of scRNA-seq preprocessing and data analysis. However, more advanced analyses can also be conducted, including:

- **Differential expression (DE) analysis**. Once clusters are identified, we can perform differential expression analysis, akin to the DE analysis covered in our previous discussions on bulk data. This analytical approach yields a list of genes exhibiting differential expression when compared against one or more clusters.
- **Trajectory inference**. Following the identification of clusters, we can infer the connection between the clusters (in that sense this step is an enhanced clustering analysis). By looking at individual cells in the vicinity of a given cell, the algorithm tries to infer the connectivity graph among different cells and different clusters. For instance, if our dataset encompasses differentiating cells, this analysis can pinpoint clusters denoting different stages of the differentiation. Furthermore, this process can enable us to decipher the **trajectory's direction** between these stages. This valuable information aids in monitoring the gene expression changes of specific genes at different points of the differentiation process.
- **Gene dynamics**. Despite scRNA-seq experiments providing a snapshot of gene expressions for each cell at a single timepoint, computational techniques allow us to deduce the dynamics of **how cells alter their gene expressions over time**, including the direction of these changes. This form of analysis was previously unattainable prior to the advent of scRNA-seq technology.
- **Compositional analysis**. In scenarios involving distinct samples, a twofold approach can be employed. Initially, basic analysis is conducted to uncover clusters within each sample. Subsequently, compositional analysis is undertaken to unveil dissimilarities in the composition of two different samples. For example, when dealing with two tumor samples from separate individuals afflicted by similar cancers, this form of analysis aids in identifying the similarities and differences in terms of compositional makeup of these samples. For example, we can assess whether one sample has more immune cells compared to the other sample. It is noteworthy that this type of analysis often necessitates the utilization of an array of computational methods and algorithms. Ensuring comparability among different samples poses a challenge that must be addressed, reflecting the intricate nature of this analysis.

Note

Additional data analysis methodologies are currently under development. Notably, there is a surge in approaches aimed at integrating diverse data types, such as combining scRNA-seq and scATAC-seq data.

Note

Computational methods have been developed to deconvolute the bulk data. For instance, in the case of bulk RNA-seq data, tools such as CIBERSORT use bulk RNA-seq data together with other reference data to estimate the abundances of cell types in a mixed cell population (Newman et al., 2015). See Cobos et al., 2020 for a benchmark of such tools.

The reference data may have been obtained from SCS approaches.

As there are a lot of publicly available bulk RNA-seq data, these methods enable us to extract more information from the bulk data.

Databases

Due to the advancements in single-cell sequencing technologies, there has been a significant increase in the production of single-cell data. A subset of this data is now accessible to the public through specialized databases like the Single Cell Expression Atlas.

Notes

- Majority of the content in this chapter is based on a lecture given by Tampere University Lecturer Juha Kesseli on 2021.

Further reading

- Life cell by cell: Introduction to Single Cell Expression Atlas a 41-minute video about Single Cell Expression Atlas.
- Current best practices in single-cell RNA-seq analysis: a tutorial (Luecken M. and Theis F., 2019)
- The Single-cell best practices is an online book (an ongoing project) with the aim of gathering information about best practices in single-cell data analysis.
- A hitchhiker's guide to single-cell transcriptomics and data analysis pipelines
- Dimensionality reduction for visualizing single-cell data using UMAP

> - Initialization is critical for preserving global data structure in both t-SNE and UMAP

⁂

Glossary

> **Note**
>
> This glossary encompasses a selected set of terms.

A

Adapter. Short single-stranded DNA sequence that is attached to both ends of a DNA fragment to be sequenced via Illumina sequencing machines. They help the DNA fragments to be attached to the Illumina sequencing flow cells.

Algorithm. It is a step by step list of directions used to solve a problem (e.g., a logical or a mathematical problem). For example, an algorithm to find out whether a number is odd (e.g., `3`) or even (e.g., `2`).

Alternative splicing. The process that allows a single gene to code for multiple proteins. This is done by keeping or excluding an exon (or exons) from the final, processed mRNA. Also known as *differential splicing*.

Amino acid. A building block of a protein. There are 20 naturally occurring amino acids.

B

No terms yet.

C

Coding sequence (CDS). A portion of a DNA sequence that is transcribed (i.e., to copied) into mRNA during transcription and then translated into protein.

Codon. A nucleotide triplet (of type DNA or RNA) that codes for amino acids, translation start signal (start codon), or translation stop signal (stop codon).

Complementary DNA (cDNA). A DNA sequence made from an mRNA that is produced by the reverse transcriptase enzyme. Since an mRNA does not include introns, cDNA will not have introns either.

D

DNA synthesis. The natural or artificial creation of DNA.

E

Exon. A region in a gene that is coding or in other words is used when creating a protein.

F

Flow chart. It is a diagram drawn using boxes and arrows (and other geometric objects each with a particular meaning) that shows an algorithm visually.

G

Gene. A stretch of DNA sequence that contains information for encoding a gene product (RNA/protein). A gene is composed of exons, introns and other regions that used in gene regulation.

Genome. The entirety of an organism's genetic information. This encompasses all genes as well as the rest of the DNA sequence.

Genomics. A scientific field of research that studies and analyzes the complete genome of an organism.

H

High-level programming language. A programming language designed to be easy for humans to read and write (e.g., Python).

Hyperparameter. In the context of machine learning, hyperparameter refers to a parameter whose value is set before the learning process begins and used to control the learning process. Additionally, its value is not learned from the data during training.

I

Interpreter. A program that reads another program, interprets it and executes it. For example, a Python interpreter, interprets and executes a Python script.

Intron. A region in a gene that is non-coding or in other words is not used when creating protein.

J

No terms yet.

K

No terms yet.

L

Low-level programming language. A programming language designed to be easy for a computer to execute. Other synonyms are: *machine language* and *assembly language*.

M

Messenger RNA (mRNA). An RNA molecule that is synthesized during transcription and serve as templates for protein synthesis. It is also known as *mature messenger RNA*.

N

Natural language. A language spoken by a group of people that evolved naturally (e.g., Finnish).

O

Open reading frame (ORF). A reading frame containing a stretch of consecutive codons (some say around 100 codons) with no stop codon. This can be translated into protein.

P

Parsing a program or a file. To examine the contents of a program or a file and analyze the syntactic structure.

Peptide bond. A chemical bond that connects two amino acids to each other.

Phosphodiester bond. A chemical bonds that attaches two nucleotides to each other in a DNA sequence.

Poly(A) tail. A stretch of RNA that has only adenine bases.

Precursor mRNA (pre-mRNA). A type of primary transcript that becomes a messenger RNA (mRNA) after processing.

Primary transcript. A single-stranded RNA produced via transcription of DNA. A primary transcript is further processed and results in a mature RNA product (e.g., mRNAs, tRNAs, etc).

Primer. A short single-stranded nucleic acid used in the initiation of DNA synthesis.

Program. A set of instructions that specifies a computation.

Programming language. It is a type of written language that tells computers what to do (e.g., Python, assembly).

Promoter. A portion of a DNA sequence before a gene that is recognized by the RNA polymerase enzyme. After the binding of RNA polymerase, the transcription may begin.

Prompt. Characters displayed by an interpreter or a terminal indicating that it is ready to take input from the user.

Protein. A chain of amino acids where each amino acid is connected to the next via a peptide bond.

Proteome. Collection of all proteins that are encoded by the genome and expressed in a particular cell.

Q

No terms yet.

R

Reading frame. A continuous, non-overlapping set of codons in a DNA sequence.

Restriction enzyme. An enzyme that cuts DNA molecules at specific recognition sequences.

Ribosome. A macromolecular machine in living cells that synthesizes protein.

S

Script. A program written in a particular programming language stored in a file.

Splice variants. The result of alternative splicing which are proteins of different length.

Splicing. The process that generates mRNA by the removal of introns from the primary transcript and joining the exons together.

Start codon. A codon that signals the start of the translation process. It almost always codes for methionine. AUG (i.e., ATG) is the most common start codon.

Stop codon. A codon that signals the termination of the translation process. Three stop codons are UAG (i.e., TAG), UGA (i.e., TGA), and UAA (i.e., TAA).

Syntax. These are the rules that determine the structure of a program. Different languages use different syntax. For example, a language may specify a particular block of code by enclosing it inside curly brackets (i.e., {}) while another language achieves this by indentation.

T

TATA box. A type of promoter DNA sequence that indicates where transcription begins. It specifies the direction of transcription and also the DNA strand to be read.

Transcription. The process of transcribing (i.e., to copy) a DNA sequence in order to produce an RNA molecule with the help of the RNA polymerase enzyme.

Transcription factor (TF). A protein that regulates the transcription of genes. It interacts with RNA polymerase.

Transcription factor binding site (TFBS). A site where a transcription factor binds.

Transcription start site (TSS). The site in a DNA sequence where transcription starts which is at the 5' (read 5 prime) end of a gene.

Transcription terminator. The site in a DNA sequence where transcription ends.

Transcriptome. Collection of all coding and non-coding transcribed DNA sequences in an organism.

Transcriptomics. A scientific field of research that studies and analyzes the complete transcriptome of an organism.

Translation. The use of mRNA as the template in order to synthesize proteins at ribosome.

U

Untranslated region (UTR). Portion of mRNA that are non-coding or in other words is not used when creating a protein.

X

No terms yet.

Y

No terms yet.

Z

No terms yet.

Starting with numbers

3' UTR. The untranslated region downstream of the CDS. It starts immediately following the translation stop codon.

5' UTR. The untranslated region upstream of the CDS. It ends as soon as the translation start codon is reached. Important regulatory regions such as the ribosome binding site can be found in this region.

⁂

Changelog

- 22.11.2023. Chapter 10 → Updated
- 18.11.2023. Chapter 8 → Updated, added description for two histone modifications
- 03.09.2023. Added the table of content
- 03.09.2023. Chapter 1 → Programming languages → Added more information.
- 02.09.2023. Chapter 1 → Central processing unit (CPUs) → Improved the description of cache.
- 02.09.2023. Added the Changelog at the bottom of the coursebook.